TREBALL DE FI DE GRAU

Grau en Matemàtiques

Títol: Teoria de la Percolació

Autor: Aleix Casellas Comas

Director: Oriol Serra Albó

Departament: Departament de Matemàtiques (749)

Convocatòria: Juliol 2017:

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Agraïments

Agraeixo als pares, família i amics pel suport rebut en els moments que ho he necessitat. Especialment vull donar les gràcies a l'Oriol per ajudar-me a escollir el tema, per guiar-me en el procediment del treball i per estar disponible en tot moment.

II

Resum

Percolació denota una àrea de les matemàtiques inspirada en models de física estadística que intenten explicar la difusió de gasos o líquids en un medi amorf. El model matemàtic estudia l'aparició d'una component connexa infinita en un subgraf aleatori d'un graf donat en funció de la probabilitat de prendre les arestes (o vèrtexs). El treball es divideix en dues parts. En la primera es dóna una panoràmica de l'àrea i es tracten alguns dels seus resultats més cèlebres, incloent-hi el teorema de Harris-Kesten sobre el valor de la probabilitat crítica en el reticle de dimensió dos. En la segona es realitzen unes simulacions per l'estimació de les probabilitats crítiques en els reticles estudiats prèviament, així com l'estimació d'aquesta probabilitat en altres reticles pels quals no se'n coneixen uns valors teòrics exactes.

Abstract

Percolation denotes an area of mathematics inspired by statistical physics models which attempt to explain the flow of liquids or gases throught a disordered medium. The mathematical model studies the appearence of an infinite connected component in a random subgraph of a given graph depending on the probability of taking the edges (or vertices). The project is divided in two parts. In the first part we explain the basics concepts of percolation and we deal with some of the most important results, including the Harris-Kesten Theorem about the critical probability in the square lattice. In the second part we do simulations to estimate the critical probabilities in the lattices studied before, as well as to estimate this probability in another lattices for which the exact theorical values aren't known.

Índex

Introducció

Històricament, la teoria de la percolació es remunta a Flory i Stockmayer, quan durant la Segona Guerra Mundial la van utilitzar per a descriure quines molècules petites formen macromolècules cada vegada més grans si es creen més i més enllaços químics entre les molècules originals. Aquest procés de polimerització ens porta a la formació d'una xarxa de connexions químiques que abasten tot el sistema. Tot i així, es sol associar l'inici de la teoria de la Percolació a Broadbent i Hammersley [1] quan l'any 1957 van presentar per primer cop el terme i ho van tractar matemàticament. Es van preguntar si el centre d'una pedra porosa es mullava quan la submergien dins l'aigua. Per respondre a aquesta pregunta, van idear un model de percolació, consistent en punts situats com en el reticle $\mathbb{Z}^2$ i amb arestes que podien connectar els punts (vèrtexs) situats a distància 1 (és a dir, cada vèrtex podia estar connectat amb els seus veïns de dalt, baix, esquerra i dreta). Llavors, havien de mirar si existia un camí d'arestes obertes que anés d'un vèrtex de la frontera de la pedra al vèrtex central d'aquesta. Si augmentem raonablement la mida de la pedra (és a dir, el nombre de vèrtexs que hi ha) van veure que la probabilitat que el vèrtex central de la pedra es mullés, estava íntimament relacionat amb la probabilitat que aquest vèrtex fos el final d'un camí obert infinit. Així doncs, quina havia de ser la probabilitat amb la qual les arestes estaven obertes per tal que hi hagués un camí de l'origen a l'infinit?

Des d'aquells inicis fins avui en dia, la teoria de la percolació ha estat una àrea de gran interès (només cal buscar a MatSciNet la paraula 'percolation' per adonar-nos-en). D'una banda tenim el seu gran ús pràctic, per la facilitat de modelar la difusió d'un líquid o gas en un medi aleatori que s'assembli molt a la realitat i obtenir-ne prediccions molt encertades. Per exemple, al llarg de la història s'ha usat per controlar la propagació de focs, l'extensió d'epidèmies (per impedir pandèmies) o la distribució de la gasolina o petroli en una roca porosa en les reserves de petroli. Des del punt de vista matemàtic, és una àrea molt fascinant per la facilitat de presentar resultats o conjectures, i la dificultat per demostrar-les rigorosament. La teoria de percolació s'ha centrat bàsicament en l'existència i el càlcul de l'anomenada probabilitat

crítica o transició de fase. Un valor de la probabilitat per sota del qual la probabilitat de percolació (és a dir, l'existència d'aquest camí infinit) és 0 i a partir d'aquí passa a ser 1. Aquests fenòmens de transició brusca són molt freqüents en models físics i han estat profundament estudiats.

Mitjançant uns experiments de Monte Carlo de Hammersley l'any 1960 s'intuïa quina podia ser la probabilitat crítica p_c en percolació en arestes en el pla, però no va ser fins a l'any 1980 que Kesten [2] va demostrar rigorosament que aquesta probabilitat crítica és $1/2$. Ho va fer usant un resultat que havia trobat Harris l'any 1960 [3] relacionat amb una fita inferior per aquesta probabilitat crítica i un resultat de Russo [4], Seymour i Welsh [5]. Es va tardar més de 20 anys en provar l'existència i trobar el valor d'aquesta probabilitat crítica, i avui en dia encara s'han demostrat rigorosament poques probabilitats crítiques per altres reticles: el reticle triangular per percolació en vèrtexs, el reticle triangular per percolació en arestes i el reticle hexagonal per percolació en arestes. La importància de la teoria de la percolació avui en dia queda reflectida amb el fet que l'any 2010 la Unió Matemàtica Internacional (IMU) va atorgar la medalla Fields al matemàtic rus Stanislav Smirnov pel seu estudi en aquesta àrea titulat "*the proof of conformal invariance of percolation and the planar Ising model in statistical physics*".

L'objectiu d'aquest treball és presentar la teoria de percolació en els reticles $\mathbb{Z}^2$ i reticles triangulars i fer simulacions per obtenir aproximacions numèriques de les probabilitats crítiques en diversos models. Ens centrarem sobretot en el resultat més important de la teoria de percolació, el Teorema de Harris-Kesten, que és el que determina la probabilitat de percolació en arestes en $\mathbb{Z}^2$. També demostrarem l'existència i trobarem el valor de la probabilitat crítica per percolació en vèrtexs en el reticle triangular, i donarem els resultats i arguments necessaris per trobar el valor de la probabilitat crítica en percolació en arestes en aquest reticle. Després, tirarem enrere en el temps i ens situarem en els anys 60 (tot i que farem ús d'un ordinador actual) i al no existir tots aquests resultats teòrics que es coneixen avui en dia farem simulacions dels fenòmens de percolació en diversos models. Amb aquestes simulacions obtindrem estimacions numèriques de les probabilitats crítiques en aquests reticles que confirmaran els resultats teòrics en els casos que es coneixen.

El treball estarà dividit en cinc capítols, que explicarem a continuació.

Per l'obtenció dels tres primers capítols, ens hem basat en el llibre *Percolation* de Bollobás i Riordan [6]. Aquests autors han contribuït de manera substancial en el desenvolupament de la teoria de la percolació i aquest text recull el seu punt de vista sobre el tema. Les simulacions estan basades en una pràctica realitzada en el material del curs *Algorithms, Part I* [7] de Kevin Wayne i Robert Sedgewick produït per la Universitat de Princeton. En

aquest curs hi ha una pràctica en Java que justament tracta el problema de percolació i vaig matricular-me a aquest curs per a produir els algorismes de simulació d'aquest treball. Tots els algorismes que s'hi exposen estan inspirats en el material d'aquest curs.

El primer és un capítol introductori, on presentem els conceptes i definicions bàsiques de la teoria de la percolació, així com uns resultats probabilístics que ens seran útils posteriorment. Primer de tot presentarem el model de percolació, ja sigui el graf o mesura de probabilitat usats, i el concepte de probabilitat de percolació (íntimament relacionada amb probabilitat crítica). Després, exposarem aquestes eines probabilístiques, consistents en la llei 0-1 de Kolmogorov, el Lema de Harris [3], la fórmula de Margulis-Russo [8] [4] i un resultat de Friedgut i Kalai [9], així com tots els conceptes i definicions necessaris per anunciar-los. El Lema de Harris segurament és el resultat més utilitzat en tot el treball, i consisteix en la correlació positiva (o negativa) d'esdeveniments creixents i decreixents. La fórmula de Margulis-Russo ens dóna una equivalència entre la influència d'una aresta en un esdeveniment i la derivada de la probabilitat que aquest esdeveniment succeeixi. Finalment, el resultat de Friedgut i Kalai troba una fita inferior per la suma de la influència de les arestes d'un esdeveniment.

El segon capítol és, segurament, la part més interessant del treball. Es presenten tots els conceptes i resultats necessaris per demostrar rigorosament el Teorema de Harris-Kesten. D'aquest teorema n'hi ha moltes demostracions, algunes d'elles molt complicades o confuses. Com he dit abans, hem decidit seguir Bollobás Riordan [6], que presenta una prova "accessible als que no són especialistes" (com jo!). Hem ampliat els passos de les demostracions i resultats que es donaven per fàcils o trivials, i també hem explicat més detalladament el què, segons el meu punt de vista, no estava abordat amb total claredat. La idea de la demostració és passar de l'existència d'un camí que creua un rectangle finit, a l'existència d'aquest camí a l'infinit. El capítol està format per dos grans teoremes a demostrar: el Teorema de Harris que diu que $p_c \geq 1/2$ i el Teorema de Kesten que diu que si $p > 1/2$ aleshores existeix un clúster obert infinit amb probabilitat 1. Pel primer, començarem trobant un resultat sobre l'existència d'un creuament horitzontal en un rectangle o un creuament vertical en el dual d'aquest rectangle. Aquest ens donarà lloc a una sèrie de resultats, un dels quals és la fita de la probabilitat d'un creuament en un quadrat quan la probabilitat amb què les arestes estan obertes és $1/2$, els quals seran usats posteriorment. Després usant un teorema de Russo-Seymour-Welsh que relaciona el creuament de quadrats amb el de rectangles aconseguirem trobar una fita per un rectangle $3n \times 2n$ quan $p = 1/2$ i conseqüentment per un rectangle $6n \times 2n$. Aquesta és la clau per la demostració del Teorema de Harris. Usant dos resultats probabilístics

(fórmula de Margulis-Russo i resultat de Friedgut i Kalai) obtindrem una fita pel creuament horitzontal d'un rectangle $\rho n \times n$, i amb una construcció d'aquests rectangles aconseguirem provar, finalment, el Teorema de Kesten.

En el tercer capítol, la intenció és trobar les probabilitats crítiques en el reticle triangular, tant per vèrtexs com per arestes. En aquest cas, i a diferència del capítol anterior, no tots els resultats estan demostrats rigorosament, ja que la complexitat d'alguns d'ells excedeix l'abast d'aquest treball. Primer de tot, començarem amb un resultat que ens afirma la unicitat del clúster obert infinit en un graf infinit, localment finit, connex i de tipus finit. La demostració d'aquest teorema no la realitzarem, però sí que enunciarem i provarem els resultats necessaris per a aquesta demostració. La idea per la demostració d'aquesta unicitat és veure primer que hi ha un tipus d'esdeveniments que tenen probabilitat 0 o 1, després que l'existència d'un nombre k de clústers oberts infinits és un d'aquests esdeveniments i que només pot passar l'existència de 0,1 o infinit clústers oberts infinits. Finalment es veuria que no podem tenir 3 o més clústers oberts infinits. Per la demostració de la probabilitat crítica en percolació en vèrtexs en el reticle triangular, primer demostrarem la probabilitat en aquest reticle pel cas finit, que queda representat per un rombe. Després, usant un resultat de Menshikov [11] que diu $p_H = p_T$ i el decaïment exponencial del radi (és a dir, la distància màxima a la qual arriba el clúster obert infinit) per percolació en vèrtexs, juntament amb el resultat del cas finit serem capaços per demostrar que la probabilitat crítica és $1/2$. En l'últim apartat només donarem una idea de com s'arriba a què la probabilitat de percolació en arestes en el reticle triangular és $2sin(\pi/18)$.

En el quart capítol, intentarem i esperem aconseguir una estimació de les tres probabilitats crítiques demostrades (o exposades) prèviament, així com la probabilitat crítica en percolació en vèrtexs en $\mathbb{Z}^2$ i les probabilitats crítiques tant per percolació en vèrtexs com en arestes en $\mathbb{Z}^3$. D'aquestes tres probabilitats crítiques només se'n coneixen unes aproximacions, així que ens basarem en aquestes per dir si els resultats obtinguts eren bons o no. Per realitzar aquestes simulacions, com he comentat abans, ens hem basat en un curs de *Coursera* anomenat *Algorithms, part I*. La primera pràctica d'aquest curs consistia a crear un primer programa que, donat un reticle $\mathbb{Z}^2$ de mida $n \times n$ amb un conjunt de vèrtexs oberts, diu si existeix un camí que creua aquest quadrat. També s'havia de crear un segon programa que estima la probabilitat crítica en aquest reticle realitzant el procés d'obrir vèrtexs fins que hi ha percolació un cert nombre de vegades. Els altres programes han sortit adaptant en menor o major mesura aquest programa, o adaptant una adaptació que havíem fet del programa inicial. En tots els casos explicarem com hem plantejat els reticles per obtenir els programes corresponents

i donarem els resultats i conclusions de les estimacions obtingudes. En els reticles plans mostrem també uns vídeos on es veu com es van obrint els vèrtexs o arestes en el respectiu reticle fins que hi ha percolació, amb reticles de diverses mides (val la pena mirar-ho!).

Finalment, el cinquè i últim capítol és simplement un annex amb el codi en Java de tots els programes que hem usat per a estimar les probabilitats crítiques en el capítol quatre.

Capítol 1

Conceptes i resultats previs

En aquest capítol l'objectiu és definir els principals conceptes relacionats amb la percolació, així com enunciar i provar (no sempre) alguns resultats probabilístics que farem servir posteriorment en els següents capítols.

1.1 Conceptes bàsics

Com hem explicat en la introducció, el concepte de percolació va sortir de voler modelar com es propagava un gas o un líquid per un medi, en concret si en submergir una pedra porosa a l'aigua el centre d'aquesta es mullava. L'objectiu era calcular la probabilitat que l'aigua penetrés fins al centre. Broadbent i Hammersley (1957) van donar un model consistent en representar aquest medi porós com el reticle $\mathbb{Z}^2$, on entre dos vèrtexs pot passar l'aigua si es troben a distància 1, és a dir, un dels dos vèrtexs es troba a sobre, sota, a l'esquerra o dreta de l'altre. En aquest treball no només estudiarem el reticle $\mathbb{Z}^2$, sinó que treballarem amb el reticle triangular, així com amb l'hexagonal (molt ocasionalment). Per tant, a l'hora de modelitzar aquest medi ho farem amb un graf qualsevol Λ.

Considerarem dos tipus de percolació: una consistent en obrir vèrtexs i l'altre en obrir arestes. Les anomenarem percolació en vèrtexs i percolació en arestes. El model de percolació consisteix a suposar que cada element del graf (vèrtex o aresta segons el cas) està 'obert' o 'present' amb una probabilitat p fixada comuna a tots i independent de la resta. L'objectiu és doncs determinar en funció de p la probabilitat que aparegui una component connexa infinita del graf amb els elements oberts. Considerarem un graf Λ amb conjunt de vèrtexs $V(\Lambda)$ i conjunt d'arestes $E(\Lambda)$.

Definició 1.1. Sigui Λ un graf amb conjunt d'arestes $E(\Lambda)$, definim una *configuració* com una funció $\omega : E(\Lambda) \to \{0, 1\}$ que envia una aresta e a

ω_e, amb $\omega_e = 1$ si l'aresta està oberta i $\omega_e = 0$ si està tancada. Escriurem $\Omega = \{0,1\}^{E(\Lambda)}$ pel conjunt de totes les configuracions.

Definició 1.2. Sigui Σ la σ-àlgebra en Ω generada pels conjunts cilíndrics $C(F,\sigma) = \{\omega \in \Omega : \omega_f = \sigma_f \text{ per } f \in F\}$, on F és un subconjunt finit de E i $\sigma \in \{0,1\}^F$. Sigui $\mathbf{p} = (p_e)_{e\in E}$, amb $0 \leq p_e \leq 1$ per tot e. La *mesura de probabilitat* $\mathbb{P}^b_{\Lambda,\mathbf{p}}$ en (Ω,Σ) està definida com

$$\mathbb{P}^b_{\Lambda,\mathbf{p}}(C(F,\sigma)) = \prod_{f\in F,\sigma_f=1} p_f \prod_{f\in F,\sigma_f=0} (1-p_f)$$

En el nostre cas, $p_e = p$ per tota aresta e i escriurem $\mathbb{P}^b_{\Lambda,p} = \mathbb{P}^b_{\Lambda,\mathbf{p}}$

Similarment, podem definir la mesura de probabilitat per percolació en vèrtexs $\mathbb{P}^s_{\Lambda,p}$.

L'objectiu principal del treball és determinar la probabilitat crítica en els diversos reticles, és a dir, aquella probabilitat on es passa de la no existència a l'existència d'un clúster obert infinit.

Definició 1.3. Denotem per $\{x \to y\}$ l'existència d'un camí obert que va d'un vèrtex x a un altre y. Aleshores definim el *clúster obert que conté* x com $C_x = \{y \in \Lambda : x \to y\}$. El clúster obert que conté l'origen el denotarem per C_0.

Definició 1.4. Sigui C_x el clúster que conté el vèrtex x. Definim la probabilitat que C_x és infinit com $\theta_x(p) = \mathbb{P}_p(|C_x| = \infty) = \mathbb{P}_p(x \to \infty)$. També podríem definir aquesta probabilitat com $\theta_(p)$ si considerem el clúster que conté l'origen.

Per tant, el nostre objectiu és trobar $0 \leq p_c \leq 1$ tal que si $p < p_c$ tenim $\theta(p) = 0$ i si $p > p_c$ es compleix $\theta(p) > 0$; és a dir, $p_c = inf\{p : \theta(p) > 0\}$. Sabem de l'existència d'aquesta probabilitat crítica per dos motius. El primer és que si considerem dos vèrtexs x i y a distància d, aleshores $\theta_x(p) \geq p^d\theta_y(p)$ i per tant o bé $\theta_x(p) = 0$ per tot vèrtex x o bé $\theta_x(p) > 0$ per tot vèrtex x. El segon motiu és que la funció $\theta_x(p)$ és creixent en p. Denotarem $p_c = p_H$ com en el llibre de Bollobás en honor a Hammersley. Aleshores, escriurem $p^s_H(\Lambda)$ i $p^b_H(\Lambda)$ per les probabilitats crítiques en percolació en vèrtexs i arestes, respectivament.

Podem definir també un altra probabilitat crítica, p_T, relacionada amb la mida esperada de C_x. Quan $p < p_H$, sabem que el clúster C_x serà finit amb probabilitat 1, però això no vol dir que la mida esperada també ho sigui. Sigui x un vèrtex del graf, definim l'esperança de la mida del clúster C_x com $\chi_x(p) = \mathbb{E}_p(|C_x|)$. Com abans, $\chi_x(p)$ és una funció creixent per p i és finit si

i només si ho és per tots els vèrtexs. Aleshores, definim aquesta probabilitat $p_T = sup\{p : \chi_x(p) < \infty\} = inf\{p : \chi_x(p) = \infty\}$, la qual no depèn del vèrtex x. En aquest cas, la T és en honor a Temperley. Per definició tenim que $p_T \leq p_H$.

1.2 Resultats probabilístics

Definició 1.5. Sigui $(\Omega, \mathcal{F}, \mathbb{P})$ un espai de probabilitat i siguin $\{\mathcal{F}_n\}_{n\geq 1}$ una successió de σ-àlgebres independents contingudes en $\mathcal{F}$. Sigui $\mathcal{G}_n = \sigma(\bigcup_{k=n+1}^{\infty} \mathcal{F}_k)$ la σ-àlgebra que conté $\mathcal{F}_{n+1}, \mathcal{F}_{n+2}...$, definim la *$\sigma$-àlgebra cua* com $\mathcal{T} = \bigcap_n \mathcal{G}_n$. A un esdeveniment $A \in \mathcal{T}$ li direm *esdeveniment cua.*

Teorema 1.6 (Llei 0-1 de Kolmogorov)**.** *Sigui $\{\mathcal{F}_n\}_{n\geq 1}$ una successió de σ-àlgebres independents i $\mathcal{T}$ la seva σ-àlgebra cua. Sigui A un esdeveniment de $\mathcal{T}$. Aleshores o bé $\mathbb{P}(A) = 0$ o bé $\mathbb{P}(A) = 1$*

Demostració. Sigui $A \in \mathcal{T}$. Volem veure que A és independent de si mateix. Per $n \to \infty$ definim la σ-àlgebra $\mathcal{F}_\infty = \sigma(\bigcup_{n=1}^{\infty} \mathcal{F}_n)$ i observem que $A \in \mathcal{T} \subset \mathcal{G}_n \subset \mathcal{F}_\infty$.

Per hipòtesis la successió de σ-àlgebres $\mathcal{F}_n$ són independents i per tant $\mathcal{F}_n$ és independent de $\mathcal{G}_n$. Aleshores, com $A \in \mathcal{G}_n$ tenim que A és independent de $\mathcal{F}_n$ per a tota n. En conseqüència, A és independent de la unió dels $\mathcal{F}_n$, és a dir, de $\bigcup_n \mathcal{F}_n$.

Siguin $\sigma(A) = \emptyset, A, A^C, \Omega$ i $\sigma(\bigcup_n \mathcal{F}_n) = \mathcal{F}_\infty$. Com A és independent de $\bigcup_n \mathcal{F}_n$, $\sigma(A)$ és independent de $\sigma(\bigcup_n \mathcal{F}_n)$. Però A pertany a les dues σ-àlgebres i per tant A és independent de A. Aleshores,

$$\mathbb{P}(A) = \mathbb{P}(A \cap A) = \mathbb{P}(A)\mathbb{P}(A) = \mathbb{P}(A)^2$$

i per tant o bé $\mathbb{P}(A) = 0$ o bé $\mathbb{P}(A) = 1$. □

Segurament el resultat que serà més usat al llarg de tot el treball, és el Lema de Harris. Per enunciar-lo i demostrar-lo, necessitem introduir primer una sèrie de conceptes, cosa que farem a continuació.

Definició 1.7. Sigui S un conjunt, definim el *conjunt de les parts* de S, $\mathcal{P}(S)$, com el conjunt de tots els subconjunts de S. Si S és un conjunt finit amb n elements, aleshores el conjunt de les parts de S té 2^n elements.

Podem identificar $\mathcal{P}(S)$ amb el conjunt $\{0,1\}^S$ de totes les funcions $f : S \to \{0,1\}$. Aquest conjunt consisteix en, per cada element de S, triar si està en un subconjunt (f l'envia a 1) o no (f envia a 0). Per tant, si S és finit, tenim 2^n possibilitats (el 2 és per si l'element es troba o no en el subconjunt) de crear subconjunts on n és el cardinal de S.

Definició 1.8. Sigui S un conjunt finit, $S = [n] = \{1, 2, ...n\}$ amb $\mathcal{P}(n) = \mathcal{P}([n]) = \mathcal{P}(S)$. Anomenem *hipercub* (o simplement cub) a $Q^n = 2^{[n]} = \{0,1\}^n$, és a dir, el conjunt de totes les seqüències 0-1 de mida n. Sigui A un subconjunt de S, $A \subset [n]$. Aleshores aquest subconjunt és una seqüència de zeros i uns $(a_i)_{i=1}^n$ on $a_i = 1$ si $i \in A$ i $a_i = 0$ si $i \in [n] \backslash A$.

Podem considerar Q^n com un graf on els seus vèrtexs es troben a $\mathcal{P}(n)$ i on dos conjunts A, B estan units si la diferència simètrica $A \triangle B$ consisteix en un element. Per exemple, si $n = 3$ els conjunts $A = \{0, 0, 1\}, B = \{0, 1, 1\}$ estan connectats però A i $C = \{1, 1, 1\}$ no. És a dir, Q^n té conjunt de vèrtexs $\{0,1\}^n \subset \mathbb{R}^n$, i dos vèrtexs $\mathbf{a}$ i $\mathbf{b}$ estan connectats si $\mathbf{a} - \mathbf{b} = \pm \mathbf{e}_i$ per algun i, on $(\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n)$ és la base canònica de $\mathbb{R}^n$. Per tant, el graf Q^n té 2^n vèrtexs i $n2^{n-1}$ arestes.

Per $A, B \subset \mathcal{P}^n$, escriurem $A \leq B$ si $A \subset B$. Això converteix el conjunt de les parts $\mathcal{P}^n$ en un conjunt parcialment ordenat. El graf Q^n té una orientació natural: una aresta $\mathbf{ab}$ està orientada de $\mathbf{a}$ a $\mathbf{b}$ si $\mathbf{a} - \mathbf{b} = \mathbf{e}_i$, per algun $1 \leq i \leq n$. Escriurem $\mathbf{a} \leq \mathbf{b}$ si hi ha un camí orientat de $\mathbf{a}$ a $\mathbf{b}$, és a dir, si $a_i \leq b_i$ per tot i.

El següent pas és establir una mesura de probabilitat en Q^n, o més precisament en el seu conjunt de vèrtexs. Això és tan fàcil com associar a cada vèrtex de Q_n el resultat de tirar una moneda n vegades. És a dir, per tot $\mathbf{a} = (a_i)_{i=1}^n \in Q^n$ tenim que $a_i = 1$ si la i-èssima tirada surt cara i $a_i = 0$ si surt creu. Si usem una moneda sense trucar, és a dir, amb probabilitat de cara i creu igual a $1/2$, obtenim la mesura uniforme en Q^n. Aleshores per un subconjunt $A \subset Q^n$ la seva probabilitat és $\mathbb{P}(A) = |A|/2^n$.

Definició 1.9. Considerem que la probabilitat de cara i creu és diferent en cada tirada. Sigui p_i la probabilitat que surti cara a la i-éssima tirada, aleshores la probabilitat d'un esdeveniment A és

$$\mathbb{P}(A) = \sum_{a \in A} \prod_{a_i = 1} p_i \prod_{a_i = 0} (1 - p_i) = \sum_{a \in A} \prod_{a_i = 1} (p_i/(1 - p_i)) \prod_{i=1}^{n} (1 - p_i)$$

El cub Q^n amb aquesta probabilitat es diu *cub ponderat amb probabilitat* $\mathbf{p} = (p_i)_{i=1}^n$, i l'escriurem com $Q_{\mathbf{p}}^n$.

El cas que normalment considerarem és quan $p_i = p$ per a tot i. Aleshores $\mathbb{P}(A) = p^{|A|}(1-p)^{n-|A|}$ i escriurem Q_p^n en comptes de $Q_{\mathbf{p}}^n$.

Definició 1.10. Diem que un subconjunt $U \subset Q^n$ és un esdeveniment *monòtonament creixent* si $a, b \in Q^n$, $a \in U$ i $a \leq b$ implica que $b \in U$. Similarment, $D \subset Q^n$ és un esdeveniment *monòtonament decreixent* si $a, b \in Q^n$, $a \in D$ i $a \geq b$ implica que $b \in D$.

Definició 1.11. Diem que una família de conjunts $\mathcal{U} \subset \mathcal{P}(S)$ és *creixent* si $A \subset B \subset S$ i $A \in \mathcal{U}$ implica que $B \in \mathcal{U}$. Similarment, $\mathcal{D} \subset \mathcal{P}(S)$ és una família de conjunts *decreixent* si $A \subset B \subset S$ i $B \in \mathcal{D}$ implica que $A \in \mathcal{D}$.

Sigui $A \subset Q^n$, per $t = 0, 1$ definim $A_t = \{(a_i)_{i=1}^{n-1} : (a_1, ..., a_{n-1}, t) \in A\} \subset Q^{n-1}$. Per tant, si A és monòtonament creixent tenim $A_0 \subset A_1$ i si A és monòtonament decreixent $A_1 \subset A_0$. Sigui $Q_{\mathbf{p}'}^{n-1}$ el cub ponderat amb la mesura de probabilitat induïda per la seqüència $\mathbf{p}' = (p_i)_{i=1}^{n-1}$. Amb un abús de notació, escrivim $\mathbb{P}$ per les dues mesures de probabilitat a $Q_{\mathbf{p}}^n$ i $Q_{\mathbf{p}'}^{n-1}$. Notem que

$$\mathbb{P}(A) = (1 - p_n)\mathbb{P}(A_0) + p_n\mathbb{P}(A_1) \tag{1.1}$$

per a qualsevol conjunt $A \subset Q^n$, ja que A_0 i A_1 són els mateixos conjunts que A en excepció que el primer té l'última coordenada fixada en 0 i el segon en 1. La probabilitat que una coordenada sigui 0 és $1 - p_n$ i la probabilitat que sigui 1 és p_n i per tant la igualtat es compleix.

Amb totes aquestes definicions i enunciats previs, ja estem preparats per demostrar i provar el Lema de Harris. La prova original és de Harris [3].

Lema 1.12 (Lema de Harris)**.** *Siguin A i B subconjunts de $Q_{\boldsymbol{p}}^n$. Si els dos són conjunts creixents o els dos són conjunts decreixents aleshores*

$$\mathbb{P}(A \cap B) \geq \mathbb{P}(A)\mathbb{P}(B). \tag{1.2}$$

Si A és un conjunt creixent i B és un conjunt decreixent, aleshores

$$\mathbb{P}(A \cap B) \leq \mathbb{P}(A)\mathbb{P}(B). \tag{1.3}$$

Demostració. Provem (1.2) per inducció sobre n. Per $n = 1$ tenim que $\mathcal{P}(n) = \mathcal{P}(\{1\}) = \{\emptyset, 1\}$. Com $\mathbb{P}(\emptyset) = 0$ i $\mathbb{P}(1) = 1$ la desigualtat surt trivialment. Per tant, suposem que $n \geq 2$ i que (1.2) es compleix per $n - 1$.

Suposem que els conjunts A i B són els dos creixents o decreixents. Aleshores, o bé $A_0 \subset A_1$ i $B_0 \subset B_1$ o bé $A_1 \subset A_0$ i $B_1 \subset B_0$. En particular

$$(\mathbb{P}(A_0) - \mathbb{P}(A_1))(\mathbb{P}(B_0) - \mathbb{P}(B_1)) \geq 0. \tag{1.4}$$

Aleshores

$$\begin{aligned}
\mathbb{P}(A \cap B) &= (1 - p_n)(A_0 \cap B_0) + p_n(A_1 \cap B_1) \\
&\geq (1 - p_n)\mathbb{P}(A_0)\mathbb{P}(B_0) + p_n\mathbb{P}(A_1)\mathbb{P}(B_1) \\
&\geq (1 - p_n)\mathbb{P}(A_0)\mathbb{P}(B_0) + p_n\mathbb{P}(A_1)\mathbb{P}(B_1) \\
&\quad - p_n(1 - p_n)[(\mathbb{P}(A_0) - \mathbb{P}(A_1))(\mathbb{P}(B_0) - \mathbb{P}(B_1))] \\
&= [(1 - p_n)\mathbb{P}(A_0) + p_n\mathbb{P}(A_1)][(1 - p_n)\mathbb{P}(B_0) + p_n\mathbb{P}(B_1)] \\
&= \mathbb{P}(A)\mathbb{P}(B)
\end{aligned}$$

on hem usat (1.1) en la primera i última igualtat, la hipòtesis d'inducció en la primera desigualtat i (1.4) en la segona desigualtat (com $(1-p_n)p_n \geq 0$, la multiplicació dels dos factors és major o igual que 0).

Considerem ara que A és un conjunt creixent i B un conjunt decreixent (o al revés). Aleshores, la desigualtat (1.3) surt d'aplicar (1.2) als conjunts creixents A i $B^C = Q^n \backslash B$

$$\begin{aligned}\mathbb{P}(A \cap B) &= \mathbb{P}(A) - \mathbb{P}(A \cap B^C) \leq \mathbb{P}(A) - \mathbb{P}(A)\mathbb{P}(B^C) \\ &= \mathbb{P}(A) - \mathbb{P}(A)[1 - \mathbb{P}(B)] = \mathbb{P}(A)\mathbb{P}(B)\end{aligned}$$

□

El resultat que nosaltres normalment usarem en els capítols següents és una conseqüència immediata del resultat anterior.

Corol·lari 1.13. *Siguin $A_1, A_2, ..., A_m$ subconjunts creixents de $Q^n_{\boldsymbol{p}}$. Aleshores*

$$\mathbb{P}(A_1 \cap A_2 \cap ... \cap A_m) \geq \mathbb{P}(A_1)\mathbb{P}(A_2)...\mathbb{P}(A_m)$$

.

Demostració. Sabem que la intersecció de dos conjunts creixents és un conjunt creixent. Per tant

$$\begin{aligned}\mathbb{P}(A_1 \cap A_2 \cap ... \cap A_m) &\geq \mathbb{P}(A_1)\mathbb{P}(A_2 \cap A_3 \cap ... \cap A_m) \\ &\geq \mathbb{P}(A_1)\mathbb{P}(A_2)\mathbb{P}(A_3 \cap A_4 \cap ... \cap A_m) \\ &\geq ... \geq \mathbb{P}(A_1)\mathbb{P}(A_2)...\mathbb{P}(A_m)\end{aligned}$$

□

Un altre resultat que necessitarem més endavant és la fórmula de Margulis-Russo. Va ser provat primer per Margulis [8] i redescobert independentment per Russo [12]. Abans d'anunciar-lo, necessitem un parell de conceptes previs.

Definició 1.14. Sigui A un esdeveniment en el cub ponderat $Q^n_{\mathbf{p}}$. Donat un punt $\omega \in Q^n = \{0,1\}^n$, diem que la variable i-èssima ω_i és *pivotal* per A si exactament un dels dos punts $\omega = (\omega_1, ..., \omega_n)$ o $r_i = (\omega_1, ..., \omega_{i-1}, 1 - w_i, w_{i+1}, ..., w_n)$ és en A. El fet que la coordenada i-èssima sigui pivotal depèn del punt ω i també de l'esdeveniment A.

Definició 1.15. Definim la *influència* de la i-èssima coordenada en A com

$$\beta_i(A) = \mathbb{P}_{\mathbf{p}}(w_i \text{ és pivotal per } A)$$

Lema 1.16 (Fórmula de Margulis-Russo)**.** *Sigui A un esdeveniment creixent en el cub ponderat $Q^n_{\boldsymbol{p}}$, amb $\boldsymbol{p} = (p_1, ..., p_n)$. Aleshores*

$$\frac{\partial}{\partial p_i}\mathbb{P}_{\boldsymbol{p}}(A) = \beta_i(A).$$

En particular

$$\frac{\mathrm{d}}{\mathrm{d}p}\mathbb{P}_p(A) = \sum_{i=1}^{n} \beta_i(A)$$

Demostració. Per fer la demostració, n'hi ha prou amb veure-ho per $i = n$. Donat un punt $\mathbf{x} = (x_1, ..., x_k) \in Q^k$, on $k \leq n$. Escrivim

$$\mathbf{p}^{\mathbf{x}} = \prod_{i:x_i=1} p_i \prod_{i:x_i=0} (1 - p_i)$$

Notem que si $k = n$, aleshores $\mathbb{P}_{\mathbf{p}}(\{\mathbf{x}\}) = \mathbf{p}^{\mathbf{x}}$. Per $\mathbf{x} \in Q^{n-1}$ definim $\mathbf{x}_+ = (x_1, ..., x_{n-1}, 1)$ i $\mathbf{x}_- = (x_1, ..., x_{n-1}, 0)$. Observem que $\mathbf{x}_+, \mathbf{x}_- \in Q^n$ i que $\mathbf{p}^{\mathbf{x}_+} + \mathbf{p}^{\mathbf{x}_-} = \mathbf{p}^{\mathbf{x}}$.

Sigui $A \subset Q^n$ un conjunt creixent. Aleshores definim

$$A_a = \{\mathbf{x} \in Q^{n-1} : \mathbf{x}_+ \in A, \mathbf{x}_- \in A\}$$

i

$$A_b = \{\mathbf{x} \in Q^{n-1} : \mathbf{x}_+ \in A, \mathbf{x}_- \notin A\}$$

Observen que

$$\mathbb{P}_{\mathbf{p}}(A) = \sum_{\mathbf{x}\in A_a} (\mathbf{p}^{\mathbf{x}_+} + \mathbf{p}^{\mathbf{x}_-})) + \sum_{\mathbf{x}\in A_b} \mathbf{p}^{\mathbf{x}_+} = \sum_{x\in A} \mathbf{p}^{\mathbf{x}} + p_n \sum_{\mathbf{x}\in A_b} \mathbf{p}^{\mathbf{x}}$$

Aleshores

$$\frac{\partial}{\partial p_n}\mathbb{P}_{\mathbf{p}}(A) = \sum_{\mathbf{x}\in A_b} \mathbf{p}^{\mathbf{x}}$$

En el punt $\mathbf{x}' = (\mathbf{x}, x_n)$, la n-èssima coordenada és pivotal si i només si $\mathbf{x} \in A_b$. Per tant, aquesta última expressió és exactament $\beta_n(A)$ i obtenim

$$\frac{\partial}{\partial p_n}\mathbb{P}_{\mathbf{p}}(A) = \beta_n(A)$$

□

L'últim resultat d'aquest capítol és un enunciat per Friedgut i Kalai [9]. En aquest cas, només l'enunciarem i no el provarem.

Teorema 1.17. *Sigui A un subconjunt del cub ponderat $Q_{\boldsymbol{p}}^n$ amb $\mathbb{P}_{\boldsymbol{p}}(A) = t$. Si $\beta_i(A) \leq \delta$ per a tota i, aleshores*

$$\sum_{i=1}^{n} \beta_i(A) \geq ct(1-t)\log(1/\delta)$$

on $c > 0$ és una constant.

Capítol 2

Teorema de Harris-Kesten

En aquest capítol, el nostre objectiu és determinar $p_H = p_H^b(\mathbb{Z}^2)$, és a dir, la probabilitat crítica per percolació en arestes en el reticle $\mathbb{Z}^2$. Per això, necessitem una sèrie de lemes i teoremes previs, necessaris per a la demostració del teorema que prova aquest resultat. Primer de tot, vegem d'on surt la intenció de demostrar aquest resultat, i els descobriments que es van anar fent fins a donar-ne una demostració rigorosa. Des dels primers articles sobre la teoria de percolació a finals dels anys 50, determinar aquesta probabilitat crítica va ser una de les grans preguntes a resoldre. Els experiments de Monte Carlo de Hammersley van conjecturar que aquesta probabilitat crítica podria ser $1/2$. El primer gran descobriment va ser realitzat per Harris [3] que va provar que $p_H \geq 1/2$. Juntament amb els experiments de Monte Carlo, Hammersley va conjecturar que la probabilitat crítica era, de fet, $1/2$. Sykes i Essam [13] van donar una prova no rigorosa que $p_T^b(\mathbb{Z}^2) = 1/2$. El segon gran resultat va ser provar que $p_T^b(\mathbb{Z}^2) + p_H^b(\mathbb{Z}^2) = 1/2$. Ho van aconseguir demostrar independentment Russo [4] i Seymour i Welsh [5]. Finalment, va ser Kesten qui rigorosament va provar la conjectura: usant el resultat de Russo-Seymour-Welsh va trobar una demostració enginyosa i complexa de què $p_H = 1/2$. Per tant es va tardar més de 20 anys a provar un resultat que sembla, aparentment, natural. D'aquest teorema n'hi ha diverses demostracions. Nosaltres no donarem aquesta realitzada per Kesten, relacionada amb la probabilitat $p_T^b(\mathbb{Z}^2)$. També utilitzarem el resultat de Harris (1960) que diu que $p_H \geq 1/2$, però a l'hora de provar la igualtat estricta $p_H = 1/2$ utilitzarem un resultat de Bollobás i Riordan [10], que diu si $p > 1/2$ aleshores la probabilitat que hi hagi un clúster obert infinit és igual a 1. Més endavant entrarem en més detalls i explicarem pas per pas el raonament usat per obtenir aquesta probabilitat crítica.

2.1 Creuament de rectangles

El primer resultat que enunciarem i provarem, és un lema relacionat amb un creuament dins el reticle $\mathbb{Z}^2$.

Definició 2.1. El graf $\Lambda = \mathbb{Z}^2$ és aquell amb conjunt de vèrtexs $\{(a,b) : (a,b) \in \mathbb{Z}^2\}$, on dos vèrtexs x i y són adjacents si es troben a distància 1, és a dir, $d(x,y) = 1$ on $d(x,y) = |x_1 - y_1| + |x_2 - y_2|$. Per tant, cada vèrtex està connectat mitjançant una aresta e amb els seus veïns de dalt, baix, esquerra i dreta.

Definició 2.2. El *graf dual* de Λ és el reticle Λ^* amb conjunt de vèrtexs $\{(a+1/2, b+1/2) : (a,b) \in \mathbb{Z}^2\}$ (de fet és el dual topològic), on dos vèrtexs z i t són adjacents, igual que en Λ, si $d(z,t) = 1$.

Observem que hi ha una aresta e^* per cada aresta e del graf Λ, que és la que creua aquesta última; veure Figura 2.1. Si considerem la configuració $\omega \in \{0,1\}^{E(\Lambda)}$ en Λ, on $E(\Lambda)$ és el nombre d'arestes del graf, definirem una aresta e^* de Λ^* oberta si la corresponent aresta e de Λ és tancada, i e^* tancada si e és oberta. Per tant, la configuració ω determina els estats de les arestes en Λ i també els de Λ^*.

Definició 2.3. Un *rectangle* R en Λ es defineix com $R = [a,b] \times [c,d]$ on $a \leq b$ i $c \leq d$ són enters. Si prenem $k = b - a + 1$ i $l = d - c + 1$, aleshores R és un rectangle $k \times l$, on k i l són el nombre de vèrtex que el rectangle té horitzontal i verticalment respectivament. Aleshores, el rectangle R té kl vèrtexs i $2kl - k - l$ arestes. Diem que R^* és un rectangle en el dual Λ^* si $R^* + (1/2, 1/2)$ és un rectangle en Λ.

Definició 2.4. Definim el *dual horitzontal* d'un rectangle $R = [a,b] \times [c,d]$ en Λ o Λ^* com el rectangle $R^h = [a + 1/2, b - 1/2,] \times [c - 1/2, d + 1/2]$, i aquest es troba en la xarxa dual; veure Figura 2.1. Anàlogament, el *dual vertical* d'un rectangle $R = [a,b] \times [c,d]$ és el rectangle $R^v = [a - 1/2, b + 1/2] \times [c + 1/2, d - 1/2]$; veure Figura 2.2.

Observem que el dual horitzontal d'un rectangle $1 \times l$ és un rectangle buit, i el dual vertical d'un rectangle $k \times 1$ també és el rectangle buit. Si $k, l \geq 2$ el dual horitzontal d'un rectangle $k \times l$ és un rectangle $(k-1) \times (l+1)$ i el dual vertical és un rectangle $(k+1) \times (l-1)$. Trivialment tenim que $(R^h)^v = (R^v)^h = R$.

Anem ara a definir el concepte de creuament, fonamental i imprescindible per gairebé tots els enunciats i/o demostracions d'aquest capítol.

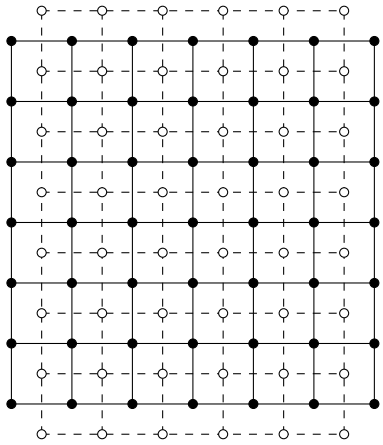

Figura 2.1: Les línies i punts negres formen un rectangle R. Expressem el seu rectangle horitzontal dual R^h amb punts blancs i línies discontínues. Si estenguéssim aquests rectangles a l'infinit, obtindríem Z^2 i el seu dual i tindríem una aresta e^* per cada aresta e.

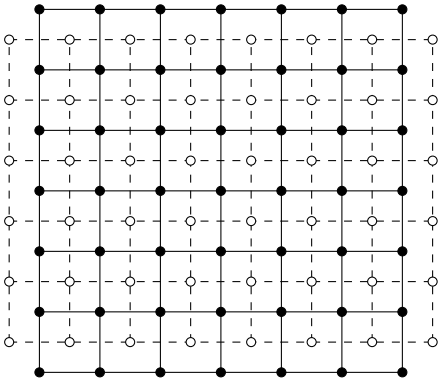

Figura 2.2: Com abans, les línies i punts negres formen un rectangle R. En aquest cas, les línies discontínues i punts blancs formen el rectangle dual vertical R^v.

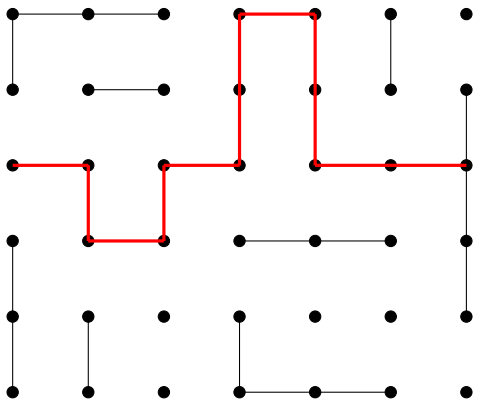

Figura 2.3: Creuament horitzontal obert d'un rectangle R, representat per les arestes de color vermell.

Definició 2.5. Donada una configuració ω, el *creuament horitzontal obert* d'un rectangle $R = [a,b] \times [c,d]$ en Λ o Λ^* és un camí obert $P \subset R$ que uneix un punt de l'esquerra del rectangle amb un punt de la dreta, és a dir, uneix (a,x) amb (b,y) $(x,y \in [c,d])$ mitjançant un conjunt d'arestes obertes de R; veure Figura 2.3. Anomenen $H(R)$ l'esdeveniment que R tingui aquest creuament. Com volem que aquest creuament horitzontal sigui mínim, el camí no contindrà cap aresta que vagi d'un vèrtex del costat esquerre de R a un altre (ni tampoc pel costat dret), és a dir, no contindrà cap aresta entre dos vèrtexs de la forma (a,x) i (a,y) (ni tampoc entre (b,x) i (b,y)). Per tant, l'esdeveniment $H(R)$ només dependrà de les arestes les quals el seu dual apareixen en R^h.

Definició 2.6. Anàlogament, $V(R)$ és l'esdeveniment que R tingui un *creuament vertical obert* i aquest només dependrà de les arestes les quals el seu dual apareixen en R^v; veure Figura 2.4.

Intuïtivament, la raó pel qual la probabilitat crítica p_H hauria de ser igual a $1/2$ és perquè per $p = 1/2$ la probabilitat que existeixi un camí obert que creua un rectangle $(n+1) \times n$ horitzontalment és $1/2$. En el següent i primer lema veurem la prova d'aquesta intuïció. De fet, veurem un cas més general (el cas concret el veiem en el corol·lari posterior). El resultat sembla bastant obvi, però no és del tot trivial de provar.

Lema 2.7. *Sigui R un rectangle de $\mathbb{Z}^2$ o el seu dual. Aleshores o bé $H(R)$ o bé $V(R^h)$ es compleix, independentment dels estats de les arestes.*

Demostració. Per realitzar aquesta demostració, considerem la següent representació del rectangle (amb arestes obertes i tancades) i el seu dual mitjançant octàgons i quadrats; veure Figura 2.5. Pintem de color negre els

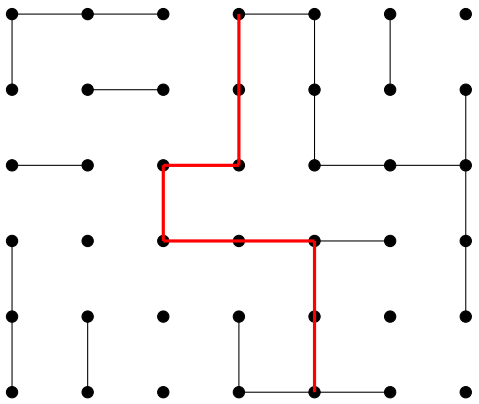

Figura 2.4: Creuament vertical obert d'un rectangle R, representat per les arestes de color vermell.

octàgons corresponents als vèrtexs del rectangle R, i de color blanc els octàgons corresponents als vèrtexs del rectangle R^h. Un quadrat és de color negre si correspon a una aresta oberta de R (tancada de R^h), i blanca si correspon a una tancada de R (oberta de R^h). Pintem de negre els quadrats (que corresponen a arestes) dels costats esquerre i dret de R per tal d'ajuntar els vèrtexs. Això no té influència en l'esdeveniment $H(R)$, ja que com hem explicat anteriorment aquest només depèn de les arestes que apareixen en R^h. Anàlogament, pintem de color blanc els quadrats dels costats superior i inferior de R^h, que pel mateix motiu no influencien en l'esdeveniment $V(R^h)$.

Observem que l'esdeveniment $H(R)$ es compleix si i només si hi ha un camí d'octàgons i quadrats negres de l'esquerra a la dreta de la figura. De la mateixa manera, $V(R^h)$ es compleix si i només si hi ha un camí d'octàgons i quadrats blancs de baix a dalt. Mirant la figura observem que no és possible que els dos esdeveniments passin simultàniament. Per exemple, en el nostre cas veiem que tenim un camí d'octàgons i quadrats negres que connecten esquerra i dreta, és a dir, l'esdeveniment $H(R)$ es compleix. Si volguéssim aconseguir $V(R^h)$, hauríem de pintar, com a mínim, un dels quadrats de color negre de color blanc. En aquest cas, pintant-ne un és suficient, però aquest quadrat està dins el camí que recórrer $H(R)$, i si el pintem de blanc $H(R)$ ja no es compliria. Suposem que el pintem blanc i aconseguim el creuament a $V(R^h)$. Com abans, el fet de pintar un o més quadrats de negre podria suposar aconseguir que es complís $H(R)$, però també que $V(R^h)$ ja no ho fes. Això no és un argument rigorós de què aquests dos esdeveniments no poden passar simultàniament, però és la idea per entendre-ho. La raó per la qual això no pot passar, és que si passés $K_{3,3}$ seria pla. Suposem que tenim quatre vèrtexs a, b, c, d que formen un cicle (un quadrat) com en la Figura 2.6.

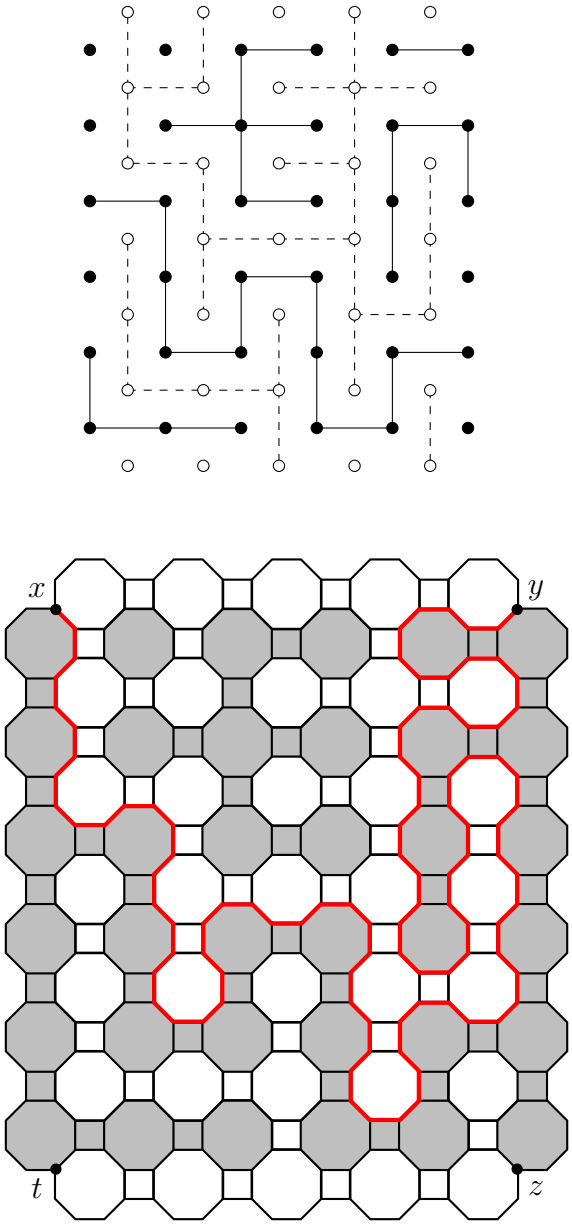

Figura 2.5: Un rectangle R i el seu dual horitzontal R^h, on les línies contínues denoten arestes obertes en R (tancades en R^h), i les línies discontínues denoten arestes obertes en R^h (tancades en R). Substituïm els vèrtexs de R per octàgons de color negre, i els vèrtexs de R^h per octàgons de color blanc. Les arestes e i e^* que connecten els vèrtexs corresponents de R i R^h estan representades pel mateix quadrat i el pintem de color negre si e està oberta en R (e^* tancada en R^h) i de color blanc altrament. Pintem de color negre els quadrats corresponents a les arestes dels costats dret i esquerra de R, i de color blanc els quadrats corresponents a les arestes de la part inferior i superior de R^h.

Afegim dos vèrtexs més e, f a dues cantonades del quadrat. Si existissin dos camins a l'interior del quadrat que connectessin a amb c i b amb d, aleshores unint e amb f per fora el quadrat tindríem que $K_{3,3}$ és un graf pla!

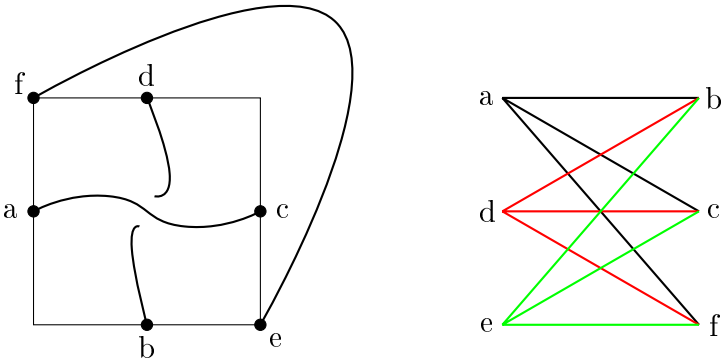

Figura 2.6: Si el cicle format per a, b, c i d conté aquests dos camins a l'interior que uneixen a i c i b i d, aleshores $K_{3,3}$ seria pla. La figura de la dreta ens mostra que la figura de l'esquerra és equivalent a $K_{3,3}$.

Sigui I el graf d'interfície, consistent en aquelles arestes dels octàgons i quadrats que separen una regió negra d'una blanca, prenent com a vèrtexs els punts finals d'aquestes arestes. Amb aquesta representació, cada octàgon està compost per 8 vèrtexs i 8 arestes, i cada quadrat per 2 vèrtexs i 2 arestes. Observem que cada vèrtex té grau 2 exceptuant els quatre vèrtexs x, y, z i t, que tenen grau 1. Per tant, el camí T que comença en x acaba en un dels vèrtexs de grau 1, és a dir, acaba en y, z o t. Sortint des de x, ens trobem que tenim una cara de color negra a l'esquerra, i una cara de color blanc a la dreta. Per com està definit I, això es complirà en tot el camí T, i per tant aquest camí només pot acabar a y o a t. Si el camí T acaba en y, com en el nostre cas, els quadrats i octàgons de la dreta de T formen un creuament horitzontal obert de R. Aquests octàgons i quadrats corresponen a un subgraf connectat S de I, que uneix els costats esquerre i dret de R. Totes les arestes de S són obertes, excepte potser algunes que estarien en els costats esquerre i dret de R (aquells quadrats els quals hem pintat de color negre només per unir els vèrtexs). Aquestes arestes no les utilitzarem per definir el graf connectat mínim P de S, ja que com hem explicat abans no són rellevants. Aquest P és un creuament horitzontal obert de R i per tant $H(R)$ es compleix. Si, per altra banda, T acabés en t, procediríem anàlogament i obtindríem un creuament vertical obert en R^h. Concloem doncs que o bé $H(R)$ o bé $V(R^h)$ es compleix.

□

Si $H(R)$ es compleix i per tant el camí T acaba en y, aleshores anomenem al camí P que hem trobat creuament horitzontal verticalment maximal. A la pràctica, pot ser molt costós i llarg trobar tot aquest graf d'interfície I. Per tant, es procedeix pas a pas: es comença des del vèrtex x i dels dos vèrtexs possibles per anar es tria aquell que deixa una part negra a la dreta, i una part blanca a l'esquerra. Es va seguint aquest procediment fins que se surt per un dels altres dos vèrtexs possible, i després es conclou si el creuament que existeix és un o l'altre. De fet, aquest algoritme és lineal en n, és a dir, eficient.

Corol·lari 2.8. *(a) Si R i R' són rectangles $k \times (l-1)$ i $(k-1) \times l$ a $\mathbb{Z}^2$, respectivament. Aleshores $\mathbb{P}_p(H(R)) + \mathbb{P}_{1-p}(V(R')) = 1$.*

(b) Si R és un rectangle $(n+1) \times n$, aleshores $\mathbb{P}_{1/2}(H(R)) = 1/2$.

(c) Si S és un quadrat $n \times n$, aleshores $\mathbb{P}_{1/2}(H(S)) = \mathbb{P}_{1/2}(V(S)) \geq 1/2$.

Demostració. (a) Si una aresta e és oberta amb probabilitat p, l'aresta e^* del dual és oberta amb probabilitat $1-p$, ja que e^* és oberta si i només si e és tancada. Pel Lema 1, o bé $H(R)$ o bé $V(R^h)$ es compleix. Per tant $\mathbb{P}_p(H(R)) + \mathbb{P}_p(V(R^h)) = 1$. R^h és un rectangle $(k-1) \times l$ en Λ^*, on les arestes són obertes amb probabilitat $1-p$. Per tant, $\mathbb{P}_p(V(R^h)) = \mathbb{P}_{1-p}(V(R'))$ i ja tenim el que volíem.

(b) Com R és un rectangle $(n+1) \times n$, R' és el mateix rectangle que R si el posem verticalment i per tant $\mathbb{P}_{1-p}(V(R')) = \mathbb{P}_p(H(R))$. Particularment si $p = 1/2$, per (a) tenim que $\mathbb{P}_{1/2}(H(R)) + \mathbb{P}_{1/2}(V(R')) = 1$. Aleshores $2\mathbb{P}_{1/2}(H(R)) = 1$, és a dir, $\mathbb{P}_{1/2}(H(R)) = 1/2$

(c) Com S és un quadrat, tenim trivialment que $\mathbb{P}_{1/2}(H(S)) = \mathbb{P}_{1/2}(V(S))$. Considerem un rectangle R $(n+1)\times n$. Aleshores $\mathbb{P}_{1/2}(H(S)) \geq \mathbb{P}_{1/2}(H(R))$, ja que necessitem un camí més curt. La conclusió s'obté novament de (a).

□

2.2 El mètode Russo-Seymour-Welsh

En aquest apartat, veurem una sèrie de resultats basats en el Teorema de Russo-Seymour-Welsh que ens relacionen el creuament de quadrats amb el creuament de rectangles, presentats en Bollobás i Riordan [10]. Començarem trobant una fita per la probabilitat d'un esdeveniment relacionat amb el creuament de rectangles, que ens servirà per trobar una fita explícita per a

un rectangle $3n \times 2n$ i posteriorment $6n \times 2n$. Aquests resultats els usarem per provar enunciats posteriors, en especial per demostrar el Teorema de Harris.

Lema 2.9. *Sigui $R = [m] \times [2n]$, $m \geq n$, un rectangle $m \times 2n$. Sigui $X(R)$ l'esdeveniment que existeixen dos camins d'arestes obertes P_1, P_2, tal que P_1 creua el quadrat $S = [n] \times [n]$ de dalt a baix, i P_2 és un camí dins de R que connecta un vèrtex de P_1 amb un del costat dret de R; veure Figura 2.7. Aleshores $\mathbb{P}_p(X(R)) \geq \mathbb{P}_p(H(R))\mathbb{P}_p(V(S))/2$*

Demostració. Suposem que $V(S)$ es compleix, és a dir, existeix un camí obert P_0 que travessa el quadrat S de dalt a baix. Aquest camí separa S en dues parts. Sigui $LV(S)$ el creuament vertical obert que està més a l'esquerra de S. Aleshores, per qualsevol valor P_1 de $LV(S)$ l'esdeveniment $\{LV(S) = P_1\}$ no depèn dels estats de les arestes que es troben a la dreta de P_1, ja que si aquest esdeveniment es compleix vol dir que aquest creuament vertical que està més a l'esquerra és P_1, i si no es compleix vol dir que hi ha un creuament obert vertical encara més a l'esquerra de P_1, que lògicament només depèn de les arestes situades a l'esquerra de P_1.

El següent pas és veure que, per a qualsevol valor P_1 de $LV(S)$, tenim

$$\mathbb{P}_p(X(R)|LV(S) = P_1) \geq \mathbb{P}_p(H(R))/2$$

Sigui P el camí obtingut unint P_1 i la seva reflexió respecte l'eix horitzontal, com en la Figura 2.7. Aleshores P creua el rectangle $[n] \times [2n]$ de dalt a baix. Per definició de $H(R)$ sabem que hi ha un camí obert P_3 que creua el rectangle R d'esquerra a dreta amb probabilitat $\mathbb{P}_p(H(R))$. Aquest camí s'ha de trobar, per força, amb el camí P. Per simetria la probabilitat que P_3 connecti amb P en algun vèrtex de P_1 és $\mathbb{P}_p(H(R))/2$.

Per simetria doncs, l'esdeveniment $Y(P_1)$ corresponent a què hi hagi un camí obert P_2 a R que connecta el costat dret de R amb un vèrtex de P_1 té probabilitat com a mínim $\mathbb{P}_p(H(R))/2$, és a dir

$$\mathbb{P}_p(Y(P_1)) \geq \mathbb{P}_p(H(R))/2$$

(Si existeix un camí que creua tot R i connecta amb P_1, P_2 existeix. Però també podria passar que existeix un camí P_2 que no creua completament R, però sí que creua una part de R fins a ajuntar-se amb P_1. Això justifica la desigualtat). Utilitzant un raonament semblant al que hem usat prèviament, es veu clarament que l'esdeveniment $Y(P_1)$ només depèn dels estats de les arestes a la dreta de P. Totes les arestes de $Y(P_1)$ en S estan a la dreta de P_1. Per tant, l'estat de les arestes de $Y(P_1)$ són independents de $\{LV(S) = P_1\}$, és a dir,

$$\mathbb{P}_p(Y(P_1)|LV(S) = P_1) = \mathbb{P}_p(Y(P_1))$$

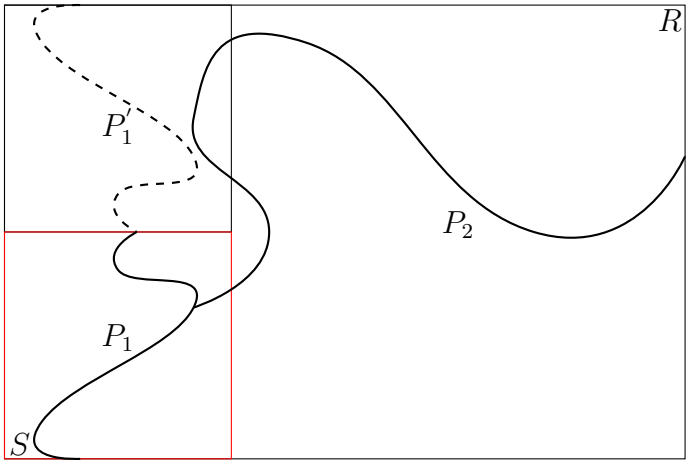

Figura 2.7: Representació d'un rectangle R amb un quadrat S (de color vermell) al seu interior. Els camins oberts P_1 i P_2 impliquen l'esdeveniment $X(R)$. El camí P està format per P_1 i la seva reflexió P_1'.

Es veu clarament que, si $Y(P_1)$ i $LV(S) = P_1$ es compleixen, també ho fa l'esdeveniment $X(R)$. Per tant

$$\mathbb{P}_p(X(R)|LV(S) = P_1) = \mathbb{P}_p(Y(P_1)|LV(S) = P_1) = \mathbb{P}_p(Y(P_1)) \geq \mathbb{P}_p(H(R))/2$$

L'esdeveniment $V(S)$ és la unió disjunta dels esdeveniments $\{LV(S) = P_1\}$, és a dir, $V(S) = \bigsqcup\{LV(S) = P_1\}$. Per tant

$$\begin{aligned}\mathbb{P}_p(X(R)|V(S)) &= \frac{\sum \mathbb{P}_p(X(R) \cap \{LV(S) = P_1\})}{\mathbb{P}_p(V(S))} \\ &= \frac{\sum \mathbb{P}_p(X(R)|\{LV(S) = P_1\})\mathbb{P}_p(LV(S) = P_1)}{\mathbb{P}_p(V(S))} \\ &\geq \frac{\mathbb{P}_p(H(R))}{2}\frac{\sum \mathbb{P}_p(LV(S) = P_1)}{\mathbb{P}_p(V(S))} = \frac{\mathbb{P}_p(H(R))}{2}\end{aligned}$$

Aleshores tenim $\mathbb{P}_p(X(R) \cap V(S))/\mathbb{P}_p(V(S)) \geq \mathbb{P}_p(H(R))/2$. Finalment, passant el denominador a l'altre banda i tenint en compte que $\mathbb{P}_p(X(R) \cap V(S)) = \mathbb{P}_p(X(R))$ obtenim el que volíem. □

Definició 2.10. Considerem un rectangle R de mida $m \times n$ en $\mathbb{Z}^2$. Definim $h_p(m,n) = \mathbb{P}_p(H(R))$ i $h(m,n) = h_{1/2}(m,n)$.

El lema anterior ens permet trobat una fita inferior de la probabilitat de creuar un rectangle $3n \times 2n$ quan $p = 1/2$.

Corol·lari 2.11. *Per a tot $n \geq 1$, $h(3n, 2n) \geq 2^{-7}$*

Demostració. Siguin R, R' dos rectangles de mida $2n \times 2n$ (és a dir, dos quadrats) i S un quadrat de mida $n \times n$ en la seva intersecció, com veiem en la Figura 2.8. Considerem l'esdeveniment $X(R)$ definit com en el Lema 2.9. Definim l'esdeveniment $X'(R')$ de la mateixa manera que $X(R)$, però en aquest cas el quadrat S es troba al cantó dret. És a dir, $X'(R')$ és l'esdeveniment que hi ha dos camins oberts tals que un creua S de dalt a baix, i l'altre uneix el costat esquerre de R' amb algun vèrtex d'aquest camí obert que creua S.

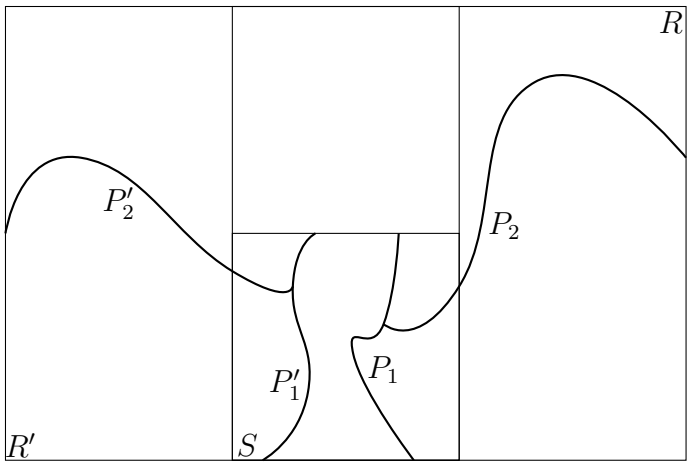

Figura 2.8: Dos rectangles R i R' de mida $2n \times 2n$ intersecats per un quadrat S de mida $n \times n$. Els camins P_1 i P_2 impliquen l'esdeveniment $X(R)$ i els camins P_1' i P_2' l'esdeveniment $X'(R')$. En aquest cas, el camí P_1' no és la reflexió de P_1.

Aplicant el Lema 2.9 als rectangles R i R' obtenim

$$\mathbb{P}_p(X'(R')) = \mathbb{P}_p(X(R)) \geq \mathbb{P}_p(H(R))\mathbb{P}_p(V(S))/2$$

Els esdeveniments $X'(R')$, $X(R)$ i $H(S)$ són creixents, ja que si es compleixen en una configuració ω, i obrim una aresta en aquesta configuració, es continuen complint. Si es compleixen els tres esdeveniments, evidentment també es compleix $H(R \cup R')$. Pel Lema de Harris, els tres esdeveniments estan correlats positivament i tenim

$$\begin{aligned} h(3n, 2n) = \mathbb{P}_{1/2}(H(R \cup R')) &\geq \mathbb{P}_{1/2}(X'(R') \cap X(R) \cap H(S)) \\ &\geq \mathbb{P}_{1/2}(X'(R'))\mathbb{P}_{1/2}(X(R))\mathbb{P}_{1/2}(H(S)) \\ &\geq \mathbb{P}_{1/2}(H(R))^2\mathbb{P}_{1/2}(V(S))^2\mathbb{P}_{1/2}(H(S))/4 \end{aligned}$$

R i S són quadrats i pel Corol·lari 2.8 (c) obtenim el que volem:

$$h(3n, 2n) \geq (1/2)^2(1/2)^2(1/2)/4 = 2^{-7}$$

□

A partir d'aquest lema, podem trobar una sèrie de resultats concrets que fiten la probabilitat que hi hagi un creuament horitzontal en diversos rectangles, molt útil per la demostració del Teorema de Harris. El punt clau es troba en aquesta facilitat per passar de quadrats a rectangles cada vegada més allargats.

Corol·lari 2.12. *Sigui R un rectangle $m_1 \times 2n$ i R' un rectangle $m_2 \times 2n$, intersecant en un quadrat S de mida $2n \times 2n$ com en la Figura 2.9. Aleshores si $m_1, m_2 \geq 2n$ tenim $h(m_1 + m_2 - 2n, 2n) \geq h(m_1, 2n)h(m_2, 2n)/2$*

Demostració. Observant la Figura 2.9 podem escriure

$$\begin{aligned} h(m_1 + m_2 - 2n, 2n) &\geq \mathbb{P}_{1/2}(H(R) \cap H(R') \cap V(S)) \\ &\geq \mathbb{P}_{1/2}(H(R))\mathbb{P}_{1/2}(H(R'))\mathbb{P}_{1/2}(V(S)) \\ &\geq h(m_1, 2n)h(m_2, 2n)/2 \end{aligned}$$

on hem usat el Lema de Harris en la segona desigualtat i el Corol·lari 2.8 (c) ($\mathbb{P}_{1/2}(V(S)) \geq 1/2$). □

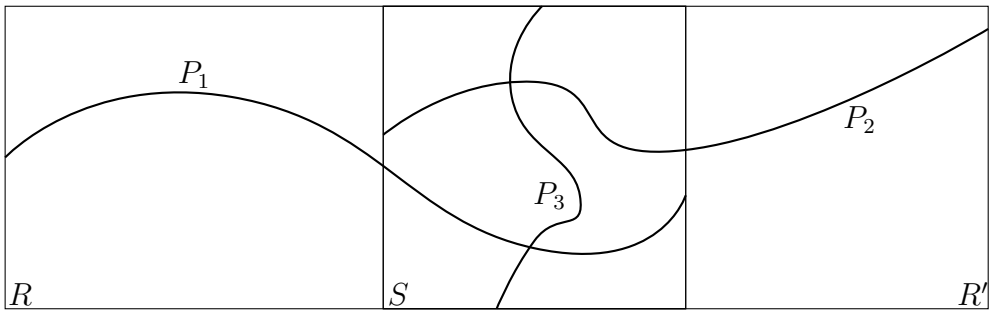

Figura 2.9: Un rectangle R de mida $m_1 \times 2n$ i un rectangle R' de mida $m_2 \times 2n$ amb $m_1, m_2 \geq 2n$ intersecats per un quadrat S de mida $2n \times 2n$. Si existeix aquests tres camins oberts (P_1 implica $H(R)$, P_2 implica $H(R')$ i P_3 implica $V(S)$) aleshores hi ha un creuament horitzontal de tota la figura.

En particular, prenent $m_1 = m, m_2 = 3n$ obtenim del Corol·lari anterior

$$h(m + n, 2n) = h(m + 3n - 2n, 2n) \geq h(m, 2n)h(3n, 2n)/2 \geq h(m, 2n)2^{-8}$$

Corol·lari 2.13. *$h(kn, 2n) \geq 2^{17-8k}$ per a tot $k \geq 3$ i $n \geq 1$*

Demostració. Vegem-ho per inducció: Si $k = 3$ tenim que $h(3n, 2n) \geq 2^{17-8\cdot 3} = 2^{-7}$. Suposem que és cert per $k = m$. Vegem que es compleix per $k = m + 1$:

$$\begin{aligned} h((m+1)n, 2n) = h(mn + n, 2n) &= h(mn + 3n - 2n, 2n) \\ &\geq h(mn, 2n)h(3n, 2n)/2 \geq 2^{17-8m}2{-}7/2 = 2^{17-8(m+1)} \end{aligned}$$

on hem usat en la segona igualtat el Corol·lari 2.12 i l'última desigualtat la hipòtesi d'inducció i el Corol·lari 2.11. □

Com $h(m, 2n+1) \geq h(m, 2n)$ (i en particular $h(kn, 2n+1) \geq h(kn, 2n)$) també tenim el cas quan n és imparell fitat. Per tant, podem afirmar que existeix una constant $h_k > 0$ tal que $h(kn, n) \geq h_k \ \forall k \geq 2$ i $n \geq 1$.

Corol·lari 2.14. *Siguin R i R' rectangles de mides $m_1 \times 2n$ i $m_2 \times 2n$ respectivament, intersecats per un rectangle $n \times 2n$ com en la Figura . Aleshores $h(m_1 + m_2 - n, 2n) \geq h(m_1, 2n)h(m_2, 2n)2^{-5}$*

Demostració. Anomenant S al quadrat $n \times n$ situat com en la Figura 2.10. Usant un raonament semblant a l'anterior (quan intersecaven en un quadrat $2n \times 2n$) tenim

$$\begin{aligned} h(m_1 + m_2 - n, 2n) &\geq \mathbb{P}(X(R) \cap X'(R') \cap H(S)) \\ &\geq \mathbb{P}(X(R))\mathbb{P}(X'(R'))\mathbb{P}(H(S)) \\ &\geq (\mathbb{P}(H(R))\mathbb{P}(V(S))/2)(\mathbb{P}(H(R'))\mathbb{P}(V(S))/2)(1/2) \\ &\geq h(m_1, 2n)h(m_2, 2n)2^{-2-2-1} \\ &= h(m_1, 2n)h(m_2, 2n)2^{-5} \end{aligned}$$

per $m_1, m_2 \geq 2n$. □

Finalment, amb tots els resultats obtinguts podem concloure que

$$h(5n, 2n) = h(3n + 3n - n, 2n) \geq h(3n, 2n)^2/2^5 \geq 2^{-19}$$

i

$$\begin{aligned} h(6n, 2n) = h(5n + 2n - n, 2n) &\geq h(5n, 2n)h(2n, 2n)/2^5 \\ &\geq 2^{-19-1-5} = 2^{-25} \end{aligned} \tag{2.1}$$

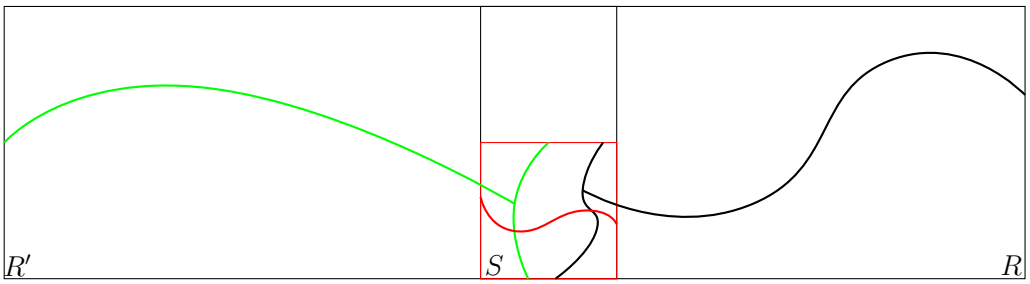

Figura 2.10: Un rectangle R de mida $m_1 \times 2n$ i un rectangle R' de mida $m_2 \times 2n$ amb $m_1, m_2 \geq 2n$ intersecats per un rectangle de mida $n \times 2n$. A la part inferior de la intersecció hi ha un quadrat S de mida $n \times n$, de color vermell. L'existència dels camins de color negre implica $X(R)$; els camins de color verd $X(R')$ i el camí de color vermell $H(S)$.

2.3 El Teorema de Harris

En aquest apartat, veurem el que segurament és el resultat més important necessari per provar que la probabilitat crítica és $1/2$. Va ser demostrat per Harris [3], però nosaltres presentarem una prova molt més senzilla i elegant, de Bollobás i Riordan [10].

Definició 2.15. El *radi* del clúster obert que conté l'origen es defineix com

$$r(C_0) = sup\{d(x,0) : x \in C_0\}$$

on $d(x,y)$ denota la distància entre dos vèrtexs de $\mathbb{Z}^2$, és a dir, $d(x,y) = |x_1 - y_1| + |x_2 - y_2|$.

Teorema 2.16. *Per percolació en arestes en* $\mathbb{Z}^2$ *tenim* $\theta(1/2) = 0$.

Demostració. Provarem

$$\mathbb{P}_{1/2}(r(C_0) \geq n) \leq n^{-c} \tag{2.2}$$

$\forall n \geq 1$, $c > 0$ constant, és a dir, una mica més del que necessitem. Com hem definit al principi del capítol, el reticle Λ correspon a la quadrícula $\mathbb{Z}^2$, i Λ^* al seu dual. Si considerem $p = 1/2$, de (2.1) tenim que la probabilitat que un rectangle $6n \times 2n$ en Λ^* tingui un creuament obert horitzontal és com a mínim 2^{-25}.

Siguin R_1, R_2 dos rectangles $6n \times 2n$ i R_3, R_4 dos rectangles $2n \times 6n$ en Λ^*, formant un anell quadrat com a la Figura 2.11. Com els esdeveniments

$H(R_i)$ són creixents, usant el Lema de Harris tenim que

$$\begin{aligned}&\mathbb{P}_{1/2}(H(R_1) \cap H(R_2) \cap H(R_3) \cap H(R_4)) \\ &\geq \mathbb{P}_{1/2}(H(R_1))\mathbb{P}_{1/2}(H(R_2))\mathbb{P}_{1/2}(H(R_3))\mathbb{P}_{1/2}(H(R_4)) \\ &\geq 2^{-25} + 2^{-25} + 2^{-25} + 2^{-25} = 2^{-100}\end{aligned}$$

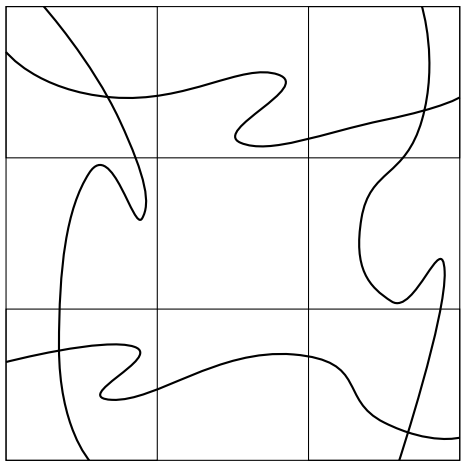

Figura 2.11: Els quatre rectangles $6n \times 2n$ formen un anell quadrat (els rectangles $3 \cdot 4^k \times 4^k$ són rectangles $6n \times 2n$). Si cada rectangle conté un creuament obert horitzontal, aleshores el centre de l'anell està envoltat per un cicle obert

És a dir, els quatre rectangles són creuats simultàniament per un camí obert diferent (o no) per cada un amb probabilitat com a mínim $\epsilon = 2^{-100}$. Si aquests quatre esdeveniments es compleixen, aleshores trivialment existeix un camí obert que envolta el centre de l'anell quadrat.

Per $k \geq 1$, sigui A_k l'anell quadrat centrat a l'origen format per dos rectangles duals $3 \cdot 4^k \times 4^k$ i dos rectangles $4^k \times 3 \cdot 4^k$ com en la Figura 2.11. Sigui E_k l'esdeveniment que A_k conté un cicle obert dual envoltant l'interior de A_k, i per tant l'origen. Aleshores $\mathbb{P}(E_k) \geq \epsilon \ \forall k$. Els anells A_k són disjunts i com a conseqüència els esdeveniments E_k són independents; veure Figura 2.12. Si l'esdeveniment E_k es compleix, aleshores com a conseqüència de la dualitat cap punt de dins de A_k es pot connectar amb un punt de fora de A_k per un camí obert a $\mathbb{Z}^2$. Analitzem aquest pas amb més detall, ja que és un argument que es fa servir sovint. Si hi hagués un camí obert que connectés el 0 amb un punt de l'exterior de A_k aleshores totes les arestes d'aquest camí estarien obertes però tancades en el dual. En algun punt,

aquest camí creuaria aquest cicle obert dual que hi ha en A_k (hem considerat els rectangles en el dual). Per tant, una o més arestes d'aquest cicle haurien d'estar tancades en el dual, cosa que ens porta a una contradicció. Així doncs, aquest camí que connecta 0 amb un punt de l'exterior de A_k no existeix i en conseqüència $r(C_0) \leq 3 \cdot 4^k/2 < 4^{k+1}$. Aleshores

$$\mathbb{P}_{1/2}(r(C_0) \geq 4^{l+1}) \leq \mathbb{P}_{1/2}(\bigcap_{k=1}^{l} E_k^C) = \prod_{k=1}^{l} \mathbb{P}_{1/2}(E_k^C) \leq (1-\epsilon)^l$$

i es compleix el que volíem veure.

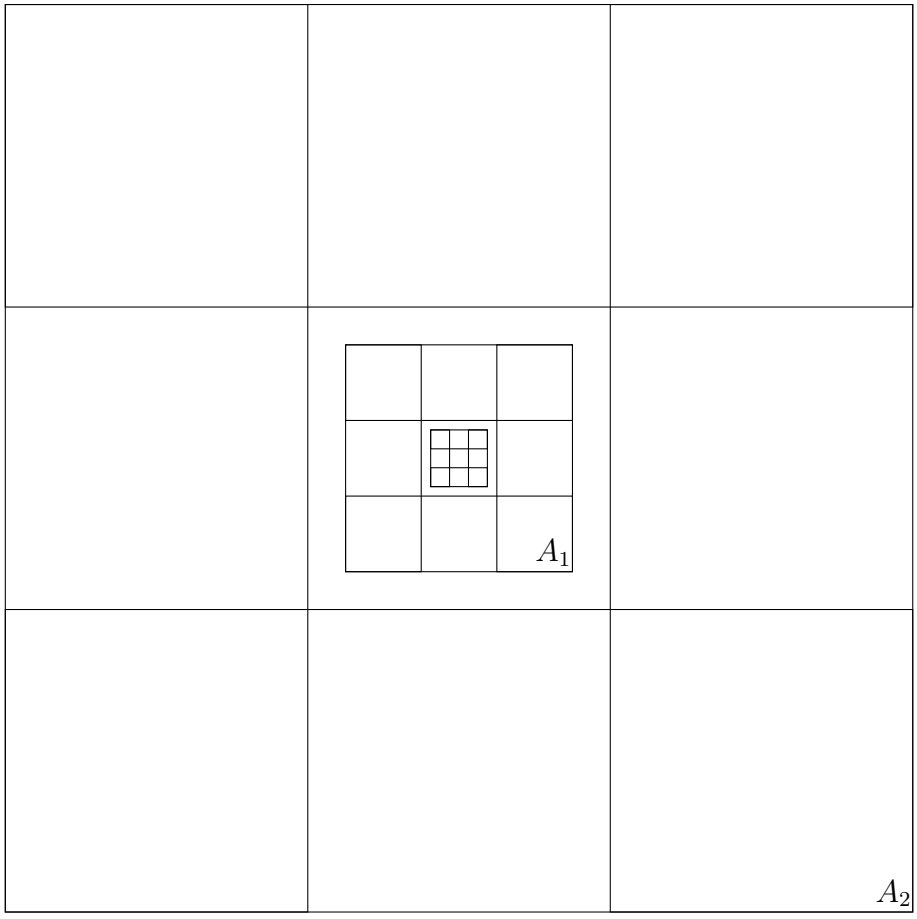

Figura 2.12: Representació dels anells quadrats A_k per $k = 0, 1, 2$. Com veiem aquests són disjunts.

Com hem dit, això és única mica més del que necessitem. Per demostrar el que ens diu el teorema, sigui n qualsevol

$$\theta(1/2) = \mathbb{P}_{1/2}(r(C_0) = \infty) \leq \mathbb{P}_{1/2}(r(C_0) \geq n) \leq n^{-c}$$

Per tant, podem concloure que $\theta(1/2) = 0$. □

Acabem de veure que $p_H \geq 1/2$. El pas més difícil d'aquest teorema, era provar el Lema 2.9, el seu conseqüent Corol·lari 2.11 i les desigualtats i corol·laris posteriors. Un cop tenim això, usant l'argument de dualitat mencionat la demostració surt fàcilment.

2.4 Una transició brusca

En aquest apartat donarem una fita inferior explícita per $h_p(m, n)$, usant el resultat de Friedgut i Kalai (Teorema 1.17) i la fórmula de Margulis-Russo (Lema 1.16).

Primer de tot, necessitem definir un parell de conceptes.

Definició 2.17. Direm que una aresta e és *pivotal* per l'esdeveniment E en una configuració ω si precisament una de les configuracions ω^+ o ω^- és en E, on ω^+, ω^- són les configuracions que coincideixen amb ω en totes les arestes diferents de e, amb e obert en ω^+, e tancat en ω^-. En altres paraules, e és pivotal si canviant l'estat de e canvia si l'esdeveniment E es compleix o no.

Definició 2.18. La *influència* de e en E és $I_p(e, E) = \mathbb{P}_p(e$ és pivotal per $E)$. Si E és creixent, sabem que $\omega^- \in E$ implica $\omega^+ \in E$ i per tant podem definir la influència com $I_p(e, E) = \mathbb{P}_p(\omega^+ \in E, \omega^- \notin E)$.

El següent lema correspon a una fita de la influència d'una aresta d'un rectangle R en l'esdeveniment $H(R)$.

Lema 2.19. *Sigui R un rectangle $m \times n$ en $\mathbb{Z}^2$, i sigui e una aresta en R. Aleshores*

$$I_p(e, H(R)) \leq 2\mathbb{P}_{1/2}(r(C_0) \geq min\{m/2 - 1, (n-1)/2\}) \tag{2.3}$$

per a tot $0 < p < 1$.

Demostració. Suposem que una aresta e de R és pivotal per l'esdeveniment creixent $H(R)$ a la configuració ω, és a dir, $\omega^+ \in H(R)$ i $w^- \notin H(R)$. Això vol dir que en la configuració ω^+ hi ha un creuament obert horitzontal de R, i aquest creuament utilitza l'aresta e, ja que en ω^- aquest creuament ja no hi és; veure Figura 2.13. Per tant un extrem de e és unit amb el costat esquerre de R per un camí obert i l'altre extrem de e és unit al costat dret de R per un camí obert a la configuració ω. Així doncs, com a mínim un extrem de e és el començament d'un camí obert de llargada mínima $m/2 - 1$ i per tant

$$I_p(e, H(R)) \leq 2\mathbb{P}_p(r(C_0) \geq m/2 - 1) \tag{2.4}$$

Aquesta desigualtat surt de suposar que e és pivotal, i en conseqüència existeix aquest camí obert. El 2 surt de la possibilitat que cada un dels extrems de l'aresta e tingui aquest camí de llargada mínima $m/2 - 1$.

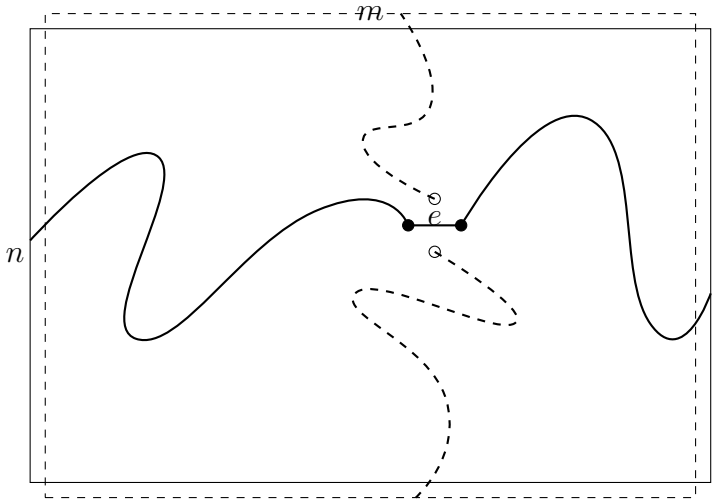

Figura 2.13: Un rectangle $m \times n$ (línies contínues) i el seu dual horitzontal de mida $(m-1) \times (n+1)$ (línies discontínues). L'aresta e és pivotal per l'esdeveniment $H(R)$. L'aresta e^* (seria la que uneix els dos cercles blancs) és pivotal per $V(R^h)$

Pel Lema 2.7, sabem que o bé $H(R)$ o bé $V(R^h)$ es compleix. En el nostre cas, com hem suposat que e es pivotal tenim que $\omega^- \notin H(R)$ i per tant s'ha de complir que $\omega^- \in V(R^h)$. Com e és pivotal també tenim $\omega^+ \in H(R)$ i per tant $\omega^+ \notin V(R^h)$. Aleshores en ω^- hi ha un camí obert dual que creua R^h verticalment fent servir e^* (l'aresta dual de e) i per tant un extrem de e^* és el començament d'un camí obert de llargada mínima $(n+1)/2 - 1 = (n-1)/2$ en la configuració ω. Considerant que en el dual les arestes estan obertes amb probabilitat $1-p$, tenim que

$$I_p(e, H(R)) \leq 2\mathbb{P}_{1-p}(r(C_0) \geq (n-1)/2) \tag{2.5}$$

Per a qualsevol a l'esdeveniment $r(C_0) \geq a$ és creixent, ja que si aquest radi és més gran que un cert valor i obrim una aresta que estava tancada, continuarà sent més gran que aquest valor. Per tant $\mathbb{P}_p(r(C_0) \geq a)$ és una funció creixent de p i podem fitar aquesta funció de la següent manera si $p \in [0, 1/2]$:

$$\mathbb{P}_p(r(C_0) \geq a) \leq \mathbb{P}_{1/2}(r(C_0) \geq a)$$

.

Si e és pivotal, hem vist que un dels dos camins existirà, i per tant $r(C_0) \geq m/2 - 1$ o bé $r(C_0) \geq (n-1)/2$. Agafant el mínim, $r(C_0) \geq min\{m/2 - 1, (n-1)/2\}$ sempre es complirà i

$$\mathbb{P}_p(r(C_0) \geq m/2 - 1) \leq \mathbb{P}_p(r(C_0) \geq min\{m/2 - 1, (n-1)/2\})$$

i també

$$\mathbb{P}_{1-p}(r(C_0) \geq (n-1)/2) \leq \mathbb{P}_{1-p}(r(C_0) \geq min\{m/2 - 1, (n-1)/2\})$$

Aleshores (2.3) surt de (2.4) per $p \leq 1/2$ i de (2.5) per $p \geq 1/2$.

□

Un cop tenim aquest resultat, ja estem capacitats per trobar aquesta fita inferior de $h_p(m,n)$. És l'últim pas que ens falta per ser capaços, finalment, de demostrar quin és el valor de la probabilitat crítica per percolació en arestes en $\mathbb{Z}^2$.

Lema 2.20. *Sigui $p > 1/2$ i sigui $\rho > 1$ un enter fixat. Aleshores existeixen constants $\gamma = \gamma(p) > 0$ i $n_0 = n_0(p,\rho)$ tal que*

$$h_p(\rho n, n) \geq 1 - n^{-\gamma}$$

per a tot $n \geq n_0$.

Demostració. Sigui R un rectangle $\rho n \times n$. En la secció 3.1 hem vist que $h(kn, n) \geq h_k$. D'aquí tenim que

$$\mathbb{P}_{1/2}(H(R)) \geq h_\rho \tag{2.6}$$

per alguna constant $h_\rho > 0$ que només depèn de ρ. Del Lema 2.19 i (2.2) per $n \geq 2$ tenim

$$I_{p'}(e, H(R)) \leq 2\mathbb{P}_{1/2}(r(C_0) \geq min\{m/2 - 1, (n-1)/2\}) \leq n^{-a} = \delta$$

per a qualsevol aresta e de R i $p' \in [1/2, p]$ on $a > 0$ és una constant (prenem $p' \in [1/2, p]$ perquè només necessitem veure què passa en aquest interval pel Teorema de Kesten).

Definim $f(p') := \mathbb{P}_{p'}(H(R))$. Pel teorema 1.17 tenim que

$$\sum_{e \in H(R)} I_{p'}(e, H(R)) \geq c f(p')(1 - f(p')) \log(1/\delta)$$

per a tot $p' \in [1/2, p]$, on c és una constant.

Pel Lema 1.16, el sumatori de sobre és exactament la derivada de $f(p')$ respecte de p'. Aleshores, escrivint $g(p') = \log(f(p')/(1-f(p')))$ obtenim

$$\begin{aligned}\frac{d}{dp'}g(p') &= \frac{1}{f(p')/(1-f(p'))}(\frac{d}{dp'}f(p')\cdot(1-f(p'))\\ &\quad - f(p')\frac{d}{dp'}(1-f(p')))\frac{1}{(1-f(p'))^2}\\ &= \frac{1}{f(p')(1-f(p'))}\frac{d}{dp'}f(p')\\ &\geq c\log(1/\delta)\\ &= ca\log n\end{aligned}$$

El següent pas és veure que $g(1/2)$ està fitada inferiorment. Tenim que $g(1/2) = \log(f(1/2)/(1-f(1/2)))$ on $f(1/2)$ està fitada inferiorment per una constant que depèn de ρ per (2.6). En conseqüència, $1-f(1/2)$ està fitada superiorment i per tant $f(1/2)/(1-f(1/2))$ també està fitada inferiorment. El logaritme d'una cosa fitada està fitada, i podem concloure que $g(1/2)$ està fitada inferiorment.

Prenent $n_0(p,\rho)$ suficientment gran, per $n \geq n_0(p,\rho)$ i usant el Teorema del valor mitjà ($f(x)-f(y) = f'(z)(x-y)$, $z \in (x,y)$) tenim

$$g(p) \geq ac(p-1/2)\log n + g(1/2) \geq ac(p-1/2)\log n$$

Per tant

$$\begin{aligned}g(p) &= \log(f(p)/(1-f(p))) = \log(h_p(\rho n,n)/(1-h_p(\rho n,n)))\\ &\geq ac(p-1/2)\log n\end{aligned}$$

És a dir

$$h_p(\rho n,n)/(1-h_p(\rho n,n)) \geq n^{ac(p-1/2)}$$

Aleshores tenim que existeix γ que depèn de p tal que

$$h_p(\rho n,n) \geq 1-n^{-\gamma}$$

□

2.5 Teorema de Kesten

En aquest apartat, veurem l'últim teorema que ens falta per demostrar que $p_H = 1/2$. Aquest darrer teorema es prova fàcilment tenint el Lema 2.20 i usant la llei 0-1 de Kolmogorov (Teorema 1.6).

Sigui E_∞ l'esdeveniment que hi ha un clúster obert infinit. Observem que $\mathbb{P}_p(E_\infty) \geq \theta_x(p) > 0$ per algun vèrtex x i si $\theta(p) > 0$ aleshores $\mathbb{P}_p(E_\infty) > 0$.

Teorema 2.21. *Per percolació en arestes en* $\mathbb{Z}^2$*, si* $p > 1/2$ *aleshores* $\mathbb{P}_p(E_\infty) = 1$.

Demostració. Fixem $p > 1/2$ i prenem els mateixos $\gamma = \gamma(p)$, $n_0 = n_0(p, 2)$ que en el Lema 2.20. Sigui $n \geq n_0$ un enter que definirem posteriorment. Per $k = 0, 1, 2, \ldots$ sigui R_k un rectangle amb llargada dels costats $2^k n$ i $2^{k+1} n$, on el costat llarg és vertical si k és parell i horitzontal si k és imparell, i sigui E_k l'esdeveniment que R_k és creuat per la part llarga del rectangle per un camí obert; veure Figura 2.14. Es veu clarament que si tots els E_k es compleixen, també ho fa E_∞, ja que dos camins qualssevol de R_k i R_{k+1} sempre es tallen. Per tant, si n és suficientment gran, aleshores pel Lema 2.20

$$\sum_{k\geq 0} \mathbb{P}(E_k^C) \leq \sum_{k\geq 0} (2^k n)^{-\gamma} < 1$$

Aleshores

$$\mathbb{P}_p(E_\infty) \geq \mathbb{P}_p(\bigcap_{k\geq 0} E_k) = 1 - \sum_{k\geq 0} \mathbb{P}_p(E_k^C) > 1 - 1 = 0$$

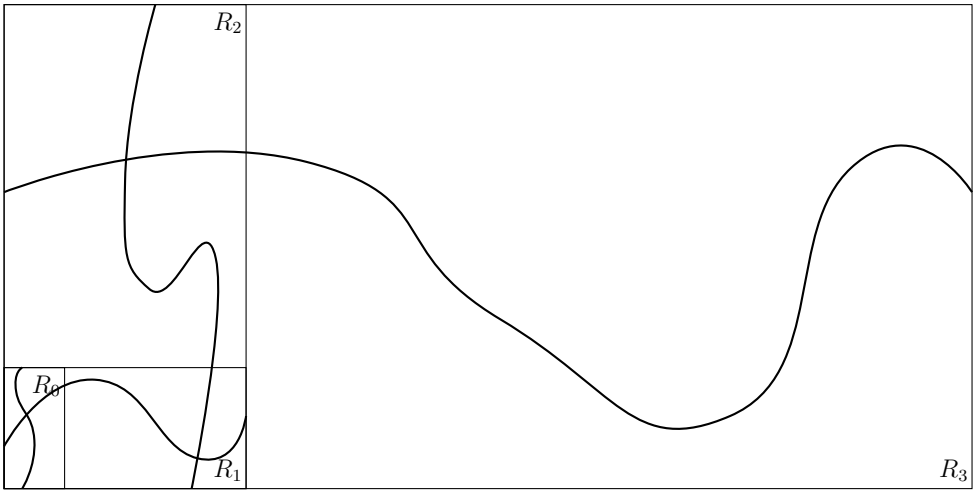

Figura 2.14: Rectangles R_k per $k = 0, 1, 2, 3$ amb els seus respectius creuaments horitzontals

Només ens falta veure que E_∞ compleix la llei 0-1 de Kolmogorov, i per tant $\mathbb{P}_p(E_\infty) = \{0, 1\}$. Com hem vist que $\mathbb{P}_p(E_\infty) > 0$ tindrem que és igual a 1. Per tant, l'únic que hem de veure és que l'esdeveniment E_∞ és independent

d'un grup finit d'arestes. Aquesta independència es veu clarament, ja que tots els vèrtexs són equivalents, i ens és indiferent mirar si existeix un clúster obert infinit que conté un vèrtex x o conté un vèrtex y. Aquest conjunt d'arestes finit el podem fitar per un quadrat (o un rectangle). Com el reticle $\mathbb{Z}^2$ és infinit, podem anar infinitament lluny d'aquest conjunt fitat d'arestes, les quals no tindran influència en l'esdeveniment E_∞.

□

Finalment, ajuntan el Teorema 2.16 i Teorema 2.21, podem concloure que $p_H(\mathbb{Z}^2) = 1/2$.

Capítol 3

El reticle triangular

En aquest capítol presentarem i estudiarem una mica el reticle triangular T amb l'objectiu de provar que $p_c^s(T) = 1/2$, és a dir, que la probabilitat crítica en percolació en vèrtexs del reticle triangular és $1/2$. Per ser capaços de provar aquest resultat, necessitem primer demostrar la unicitat dels clústers oberts infinits. També enunciarem quina és la probabilitat crítica en percolació en arestes, així com els resultats que han portat a la seva demostració. En el següent capítol simularem un parell de programes que esperem que ens donin els mateixos resultats (o una aproximació molt bona) dels teòrics.

El reticle triangular T correspon als vèrtexs situats com en $\mathbb{Z}^2$, on cada fila parella (o imparella) està desplaçada $1/2$ cap a l'esquerra (o dreta). Dos vèrtexs són adjacents si es troben a distància 1 com en la Figura 3.1.

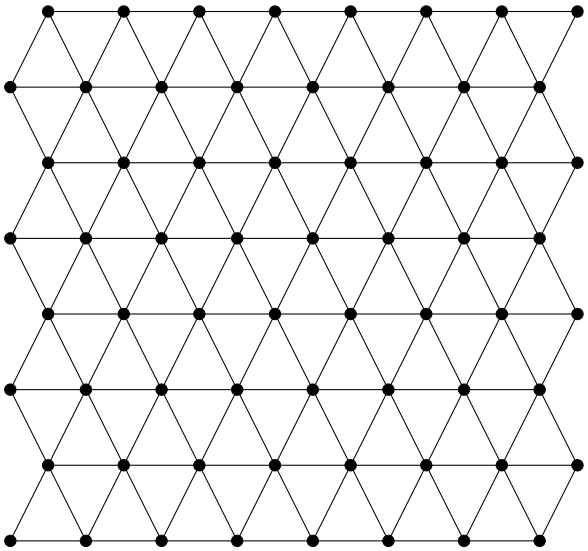

Figura 3.1: Representació gràfica del reticle triangular

3.1 Unicitat del clúster obert infinit

En aquest apartat volem veure que en percolació en vèrtexs en un graf Λ, o bé existeix un únic clúster obert infinit, o bé no n'hi ha cap. Aquest resultat va ser provat per Aizenman, Kesten, Newman [14]. Ens serà molt útil en l'apartat següent, per provar la probabilitat crítica per percolació en vèrtexs en el reticle triangular.

Sigui Λ un graf infinit, localment finit i connex. Suposem també que hi ha un nombre finit de classes d'equivalència entre vèrtexs, on dos vèrtexs x i y són equivalents si existeix un automorfisme de Λ que envia x a y. Denotarem aquesta equivalència com $x \sim y$ i direm que Λ és de tipus finit.

Considerem la mesura de probabilitat $\mathbb{P}_p = \mathbb{P}^s_{\Lambda,p}$, que és la mesura producte en l'espai producte infinit $\Omega = \{0,1\}^{V(\Lambda)}$. La σ-àlgebra Σ dels conjunts mesurables de Ω està generada pels conjunts cilíndrics $C(F,\sigma) = \{w \in \Omega : \omega_f = \sigma_f \text{ per } f \in F\}$, on F és un subconjunt finit de $V(\Lambda)$ i $\sigma \in \{0,1\}^F$.

En aquest espai, un esdeveniment és simplement un subconjunt mesurable de Ω. Per tant, qualsevol propietat que depengui d'un nombre finit de vèrtexs és un esdeveniment i la unió i intersecció d'un conjunt numerable d'esdeveniments és un esdeveniment. L'expressió $\{x \to n\}$ ens diu que existeix un camí obert de x a un altre vèrtex a distància n. Fixats una aresta x i un enter n, aquesta propietat només depèn de l'estat d'un nombre finit de vèrtexs i per tant és mesurable. Aleshores com $\{x \to \infty\}$ és la unió numerable dels esdeveniments $\{x \to n\}$, també és un esdeveniment. Per tant, els $I_k = \{\text{hi ha exactament k clústers oberts infinits}\}$ per $0 \leq k \leq \infty$ també són esdeveniments.

Definició 3.1. Un esdeveniment E és *invariant per automorfismes* si per tot automorfisme φ del graf Λ l'automorfisme induït $\varphi^* : \Omega \to \Omega$ envia E a ell mateix. En particular, I_k és un esdeveniment invariant per automorfismes per $k = 0, 1, ...\infty$.

El resultat que ve a continuació és una propietat sobre aquests esdeveniments que són invariants per automorfismes.

Lema 3.2. *Sigui Λ un graf infinit, localment finit, de tipus finit i $E \subset \Omega = \{0,1\}^{V(\Lambda)}$ un esdeveniment invariant per automorfismes. Aleshores o bé $\mathbb{P}^s_{\Lambda,p}(E) = 0$ o bé $\mathbb{P}^s_{\Lambda,p}(E) = 1$.*

Demostració. En tota la demostració, escriurem $\mathbb{P}$ en lloc de $\mathbb{P}^s_{\Lambda,p}$. Sigui x_0 un vèrtex de Λ. Com Λ és infinit, localment finit i de tipus finit, aleshores hi ha infinits vèrtexs x que són equivalents a x_0.

Prenem $\epsilon > 0$ qualsevol. L'esdeveniment E és mesurable i per tant existeixen un conjunt finit $F \in V(\Lambda)$ i un esdeveniment cilíndric E_F que només

depèn dels estats dels vèrtexs de F tals que $\mathbb{P}(E \triangle E_F) \leq \epsilon$, on $\triangle$ denota la diferència simètrica.

Definim $M = \max\{d(x,y) : y \in F\}$. El graf Λ és localment finit i per tant la bola $B_{2M}(x_0) = \{z : d(x_0,z) \leq 2M\}$ és finita. Aleshores hi ha un vèrtex x equivalent a x_0, $x \sim x_0$, tal que $d(x,x_0) > 2M$. Sigui φ un automorfisme de Λ que envia x_0 a x. Prenent $y \in F$ i usant la desigualtat triangular $d(a,c) \leq d(a,b) + d(b,c)$, amb $a = x_0$, $b = \varphi(y)$ i $c = \varphi(x_0)$ tenim

$$\begin{aligned} d(x_0, \varphi(y)) &\geq d(x_0, \varphi(x_0)) - d(\varphi(x_0), \varphi(y)) = d(x_0, x) - d(x_0, y) \\ &> 2M - M = M \end{aligned}$$

Per tant, $\varphi(y) \notin F$ i els conjunts d'arestes F i $\varphi(F)$ són disjunts. Així doncs, els esdeveniments E_F i $\varphi(E_F)$ són independents, i com $x \sim x_0$ els conjunts d'arestes E_F i $\varphi(E_F)$ són equivalents i tenim

$$\mathbb{P}(E_F \cap \varphi(E_F)) = \mathbb{P}(E_F)\mathbb{P}(\varphi(E_F)) = \mathbb{P}(E_F)\mathbb{P}(E_F) = \mathbb{P}(E_F)^2$$

Aleshores

$$|\mathbb{P}(E) - \mathbb{P}(E_F)^2| = |\mathbb{P}(E \cap E) - \mathbb{P}(E_F \cap \varphi(E_F))| \leq \mathbb{P}((E \cap E) \triangle (E_F \cap \varphi(E_F)))$$

Siguin A, B, C, D quatre conjunts. Sabem que sempre es compleix $(A \cap B) \triangle (C \cap D) \subset (A \triangle C) \cup (B \triangle D)$. Usant aquesta propietat i tenint en compte que E és invariant per automorfismes, de l'equació anterior obtenim

$$\begin{aligned} |\mathbb{P}(E) - \mathbb{P}(E_F)^2| &\leq \mathbb{P}(E \triangle E_F) + \mathbb{P}(E \triangle \varphi(E_F)) \\ &= \mathbb{P}(E \triangle E_F) + \mathbb{P}(\varphi(E) \triangle \varphi(E_F)) \\ &= \mathbb{P}(E \triangle E_F) + \mathbb{P}(E \triangle E_F) = 2 \cdot \mathbb{P}(E \triangle E_F) \leq 2\epsilon \end{aligned}$$

Aleshores, com $|\mathbb{P}(E) - \mathbb{P}(E_F)| \leq |\mathbb{P}(E \triangle E_F)| \leq \epsilon$ tenim

$$\begin{aligned} |\mathbb{P}(E) - \mathbb{P}(E)^2| &\leq |\mathbb{P}(E) - \mathbb{P}(E_F)^2| + |\mathbb{P}(E_F)^2 - \mathbb{P}(E)^2| \\ &\leq |\mathbb{P}(E) - \mathbb{P}(E_F)^2| + 2|\mathbb{P}(E_F) - \mathbb{P}(E)| \leq 2\epsilon + 2\epsilon = 4\epsilon \end{aligned}$$

Per tant, hem vist que $\mathbb{P}(E) = \mathbb{P}(E)^2$ cosa que implica que o bé $\mathbb{P}(E) = 0$ o bé $\mathbb{P}(E) = 1$. □

El resultat anterior era per un esdeveniment qualsevol invariant per automorfismes. A partir d'aquí, veurem alguns resultats per l'esdeveniment concret I_k, és a dir, l'esdeveniment hi ha k clústers oberts infinits, on $0 \leq k \leq \infty$.

Lema 3.3. *Sigui Λ un graf infinit, localment finit, de tipus finit i connex, i sigui $p \in (0,1)$. Aleshores $\mathbb{P}^s_{\Lambda,p}(\bigcup_{2\leq k<\infty} I_k) = 0$ i tenim que o bé $\mathbb{P}^s_{\Lambda,p}(I_0) = 1$ o bé $\mathbb{P}^s_{\Lambda,p}(I_1) = 1$ o bé $\mathbb{P}^s_{\Lambda,p}(I_\infty) = 1$.*

Demostració. Com abans, escriurem $\mathbb{P}$ en lloc de $\mathbb{P}^s_{\Lambda,p}$. Observem que només necessitem provar que $\mathbb{P}(\bigcup_{2\leq k<\infty} I_k) = 0$, ja que pel Lema 3.2 $\mathbb{P}(I_k) \in \{0,1\}$ per $k = 0, 1$ o ∞ i com no es poden complir els tres esdeveniments simultàniament ni dos d'ells, tindrem que un dels tres es compleix amb probabilitat 1.

Suposem que $\mathbb{P}(I_k) > 0$ per algun $2 \leq k < \infty$. Fixem un vèrtex x_0 del graf Λ. Definim $T_{n,k}$ com l'esdeveniment que I_k es compleix i cada un dels clústers oberts infinits conté un vèrtex de la bola $B_n(x_0) = \{y \in V(\Lambda) : d(x_0, y) \leq n\}$. Aquestes boles $B_n(x_0)$ cobreixen tot Λ i per tant $I_k = \bigcup_n (T_{n,k})$. Aleshores $\mathbb{P}(T_{n,k}) \nearrow \mathbb{P}(I_k)$ i existeix un n tal que $\mathbb{P}(T_{n,k}) > 0$.

Notem que podem escriure l'esdeveniment $T_{n,k}$ com la unió disjunta dels esdeveniments $T_{n,k,\mathbf{s}} = T_{n,k} \cap \{S = \mathbf{s}\}$ on $\mathbf{s} = (s_x)_{x\in B_n(x_0)} \in \{0,1\}^{V(B_n(x_0))}$ i $S = (S_x)_{x\in B_n(x_0)}$ és un vector que indica l'estat dels vèrtexs de $B_n(x_0)$. Per tant existeix un $\mathbf{s}$ tal que $T_{n.k,\mathbf{s}} > 0$.

Prenem una configuració $\omega \in T_{n,k,\mathbf{s}}$. Definim ω' com aquella configuració obtinguda de ω obrint tots aquells vèrtexs de $B_n(x_0)$ que estaven tancats. Aleshores $\omega' \in I_1$, ja que el fet d'obrir totes les arestes de $B_n(x_0)$ (com tots els clústers oberts infinits contenen un vèrtex de $B_n(x_0)$) fa que tots els clústers oberts infinits s'uneixin i en quedi només un. Per tant

$$\mathbb{P}(I_1) \geq \mathbb{P}(\{\omega' : \omega \in T_{n,k,\mathbf{s}}\}) = (p/(1-p))^c \mathbb{P}(T_{n,k,\mathbf{s}}) > 0$$

on c és el nombre de vèrtexs de $B_n(x_0)$ que estan tancats quan $S = \mathbf{s}$. El terme $(p/(1-p))^c$ surt d'obrir aquests vèrtexs que abans estaven tancats: al numerador hi ha la probabilitat d'obrir-los i al denominador s'elimina la probabilitat que estiguessin tancats.

Hem vist que si $\mathbb{P}(I_k) > 0$ per algun $2 \leq k < \infty$ llavors $\mathbb{P}(I_1) > 0$. Però pel Lema 3.2 $\mathbb{P}(I_k) = \mathbb{P}(I_1) = 1$, cosa que és impossible, ja que $I_k \cap I_1 = \emptyset$ i per tant $\mathbb{P}(I_k \cap I_1) = 0$. Així doncs, hem arribat a una contradicció pel fet de suposar que $\mathbb{P}(I_k) > 0$ per algun $2 \leq k < \infty$. Per tant, aquest I_k no existeix i podem afirmar que $\mathbb{P}(\bigcup_{2\leq k<\infty} I_k) = 0$. □

L'últim resultat necessari per a la demostració del teorema de la unicitat del clúster obert infinit és un relacionat amb els grafs finits, del que no incloem la demostració.

Lema 3.4. *Sigui G un graf finit amb k components i siguin L i $C = \{c_1, ..., c_s\}$ dos conjunts disjunts de vèrtexs de G, amb com a mínim un c_i*

a cada component de G. Siguin $m_1, ..., m_s$ enters més grans o iguals que 2. Suposem que per cada i el fet de esborrar el vèrtex c_i divideix el component que conté c_i en dos de més petits, m_i dels quals contenen vèrtexs de L. Aleshores

$$|L| \geq 2k + \sum_{i=1}^{s}(m_i - 2)$$

.

Definició 3.5. Diem que un graf infinit és *'amenable'* si $|S_n(x)|/|B_n(x)| \to 0$ quan $n \to \infty$ per a qualsevol vèrtex x, és a dir, les boles grans contenen molts més vèrtexs que la frontera de les seves esferes.

Ara ja tindríem tot el necessari per provar i demostrar aquest teorema tan important. Tot i això, nosaltres només l'enunciarem i donarem la idea de la demostració, ja que aquesta és complicada. Com hem comentat anteriorment, la prova d'aquesta demostració es troba en el llibre de Burton i Keane [15].

Teorema 3.6. *Sigui Λ un graf infinit, localment finit, de tipus finit, connex i 'amenable', i sigui $p \in (0,1)$. Aleshores o bé $\mathbb{P}^s_{\Lambda,p}(I_0) = 1$ o bé $\mathbb{P}^s_{\Lambda,p}(I_1) = 1$.*

Demostració. (esbós): La idea de la demostració és veure que $\mathbb{P}(I_\infty) = 0$, ja que pel Lema 3.3 obtindríem el que volem. Per demostrar això, es veu que la probabilitat que hi hagi com a mínim tres clústers oberts infinits és 0. Com abans, s'arriba al resultat per contradicció. □

3.2 Percolació en vèrtexs en el reticle triangular

En percolació en arestes en $\mathbb{Z}^2$, com hem vist en el capítol anterior, podem fitar un clúster obert infinit per un cicle obert en el dual $\mathbb{Z}^2 + (1/2, 1/2)$. En percolació en vèrtexs en T, es pot veure fàcilment que un clúster obert està fitat per un cicle tancat en el mateix reticle (ho veurem en la demostració del lema següent). Similarment al que el que passava en $\mathbb{Z}^2$, un camí obert en T no pot començar dins d'un cicle tancat i acabar-ne fora.

En aquest apartat volem provar que la probabilitat crítica en percolació en vèrtexs en T és $1/2$, un resultat que va demostrar Kesten [16]. Escriurem $\mathbb{P}_p$ referint-nos a $\mathbb{P}^s_{T,p}$.

Lema 3.7. *Sigui R_n un rombe en T amb n vèrtexs en cada costat, com en la Figura3.3. Sigui $H(R_n)$ l'esdeveniment tal que hi ha un creuament obert de l'esquerra a la dreta de R_n, consistent en vèrtexs oberts de T. Aleshores $\mathbb{P}_{1/2}(H(R_n)) = 1/2$ per tot $n \geq 1$.*

Demostració. Definim $V^*(R_n)$ l'esdeveniment hi ha un camí tancat a R_n que uneix la part inferior i superior de R_n. Si reflectim R_n per la seva diagonal llarga i canviem els vèrtexs oberts pels tancats com en la Figura 3.2 observem que $\mathbb{P}_p(H(R_n)) = \mathbb{P}_{1-p}(V^*(R_n))$ per qualsevol p. Si prenem $p = 1/2$, aleshores $\mathbb{P}_{1/2}(H(R_n)) = \mathbb{P}_{1/2}(V^*(R_n))$. Per tant, per demostrar aquest lema només necessitem veure que $\mathbb{P}_{1/2}(H(R_n)) + \mathbb{P}_{1/2}(V^*(R_n)) = 1$. Provarem una mica més que això: provarem que o bé $H(R_n)$ o bé $V^*(R_n)$ es compleix. Això ens recorda molt al resultat del Lema 2.7, i més si tenim en compte l'observació sobre els clústers que són envoltats per camins oberts duals i camins tancats, en $\mathbb{Z}^2$ i T respectivament. Com veurem a continuació, l'argument que farem servir és pràcticament el mateix que el que vam usar en la demostració d'aquell lema.

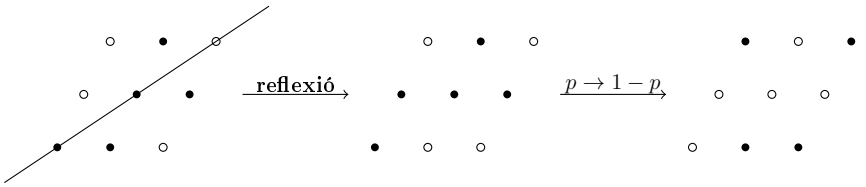

Figura 3.2: Reflexió dels vèrtexs de R_n i canvi de l'estat dels vèrtexs

Reemplacem cada vèrtex de R_n per un hexàgon, de color negre si el vèrtex està obert i de color blanc si està tancat. Afegim una línia extra d'hexàgons negres a l'esquerra i a la dreta, i una fila extra d'hexàgons blancs a sobre i sota, com en la Figura 3.3. Considerem el graf d'interfície I que té com arestes aquells costats dels hexàgons que separen una part blanca d'una negra, i com a vèrtexs els punts finals d'aquests costats. Tots els vèrtexs tenen grau 2 exceptuant x, y, z i t, que tenen grau 1. Per tant, si comencem un camí en x, aquest ha de finalitzar en un dels altres tres vèrtexs esmentats. Si ens situem en x i comencem a recórrer el graf d'interfície, veiem que sempre tenim la part dreta de color negre i l'esquerra de color blanc. Per tant, el camí P que comença en x i recorre I ha d'acabar per força en y o t. Si P acaba en y (com en el nostre cas), tenim que hi ha un creuament horitzontal en R_n format per vèrtexs oberts, és a dir, $H(R_n)$ es compleix. Si pel contrari P acabés en t, aleshores l'esdeveniment que es compliria seria $V^*(R_n)$. Veiem que els dos esdeveniments no es poden complir simultàniament, ja que per exemple en el nostre cas, si volem obtenir $V^*(R_n)$ hem de tancar com a mínim un vèrtex de R_n, i aquest vèrtex és usat per $H(R_n)$. Com en el Lema 2.7, l'argument rigorós pel qual aquests dos camins no poden ocorre simultàniament, és que si ho fessin $K_{3,3}$ seria pla.

□

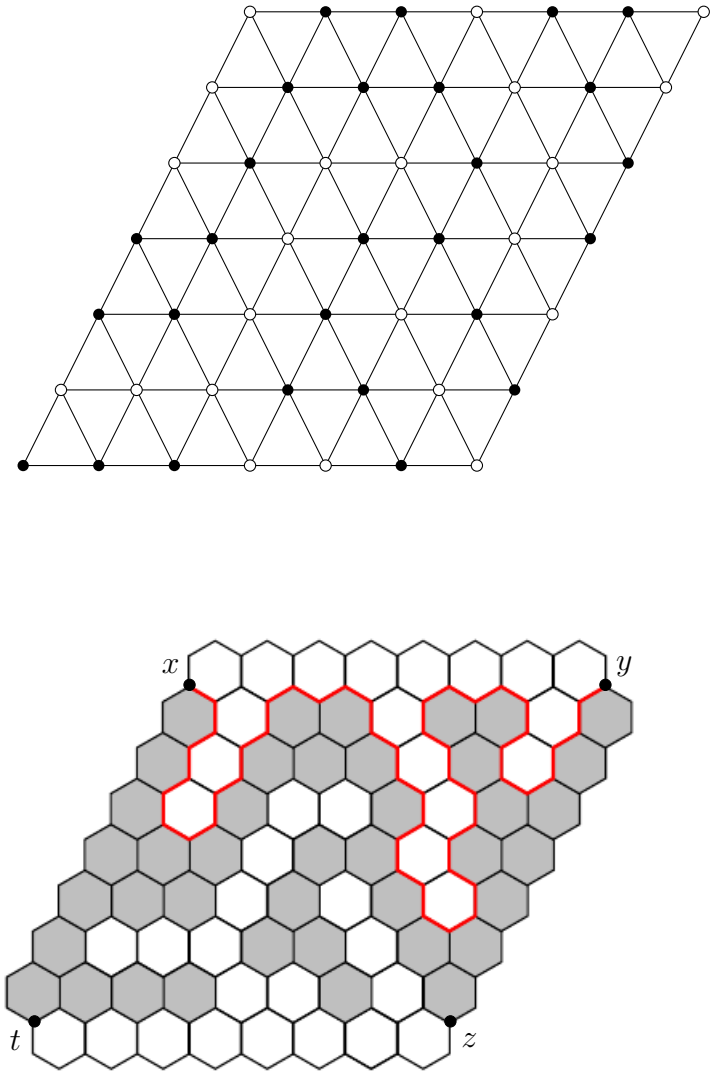

Figura 3.3: Un rombe R_n i la consegüent representació amb hexàgons. Substituïm els vèrtexs oberts per hexàgons de color negre i els tancats per hexàgons de color blanc. Afegim una fila d'hexàgons blancs a la part superior i inferior de la figura i una columna d'hexàgons negres a l'esquerra i dreta de la figura.

Per provar que la probabilitat crítica en percolació en vèrtexs és $1/2$, necessitem dos resultats previs que només enunciarem. Aquests resultats estan extrets de Bollobás i Riordan [6, Theorem 4.7, Theorem 4.9]

Teorema 3.8. *Sigui $\overrightarrow{\Lambda}$ un multigraf infinit, localment finit i orientat amb $C_{\overrightarrow{\Lambda}}$ finit i fortament connex. Si existeix una constant C tal que $|S_n^+(x)| \leq \exp(Cn/(\log(n))^3)$ per tot vèrtex x i enter n, aleshores $p_T^s(\overrightarrow{\Lambda}) = p_H^s(\overrightarrow{\Lambda})$.*

Aquest resultat, demostrat per Menshikov [11], ens diu que $p_H = p_T$. Per tant l'únic que hem de provar és que $p_c^s(T) = 1/2$, on $p_c^s(T) = p_H = p_T$.

Teorema 3.9. *Sigui $\overrightarrow{\Lambda}$ un multigraf orientat amb $C_{\overrightarrow{\Lambda}}$ finit i fortament connex. Si $p < p_T^s(\overrightarrow{\Lambda})$ aleshores existeix $\alpha > 0$ tal que $\mathbb{P}_p^s(r(C_x) \geq n) \leq e^{-\alpha n}$ per tot vèrtex x i enter $n \geq 1$.*

Ara ja podem procedir en l'enunciat i la demostració del teorema.

Teorema 3.10. *Sigui T el reticle triangular en el pla. Aleshores $p_c^s(T) = 1/2$.*

Demostració. Per demostrar aquest teorema, suposarem primer que aquesta probabilitat crítica és major que $1/2$ i després que és menor que $1/2$. En els dos casos, arribarem a una contradicció i podrem afirmar que és, efectivament, $1/2$.

Suposem que $p_c^s(T) = p_T^s(T) > 1/2$. Pel Teorema 3.9, tenim que per $p < p_T^s(T)$ $\exists \alpha > 0$ tal que $\mathbb{P}_p^s(r(C_x) \geq n) \leq e^{-\alpha n}$ per tot x i en particular $\mathbb{P}_{1/2}^s(r(C_0) \geq n) \leq e^{-\alpha n}$. Com en el lema anterior, definim R_n com el rombe en la xarxa triangular on els vèrtexs de l'esquerra estan a distància mínima $n-1$ dels de la dreta. Per tant,

$$\mathbb{P}_{1/2}(H(R_n)) \leq n\mathbb{P}_{1/2}(r(C_0) \geq n-1) \leq ne^{-\alpha(n-1)}$$

Quan $n \to \infty$ $ne^{-\alpha(n-1)} \to 0$ i per tant $\mathbb{P}_{1/2}(H(R_n)) = 0$, que és una contradicció amb el Lema 3.7. Per tant, $p_c^s(T) \leq 1/2$.

Suposem ara que $p_c^s(T) = p_H^s(T) < 1/2$, per tant $\theta(1/2) > 0$, i vegem que arribem a una contradicció. Si $p = 1/2$, sabem que existeix un únic clúster obert infinit pel Teorema 3.6. Com en altres demostracions, ens ajudarem per una representació gràfica per trobar aquesta contradicció. Sigui H_n un hexàgon amb n vèrtexs a cada costat centrat a l'origen, com en la Figura 3.4. Clarament, $T = \bigcup_n H_n$. Si n és suficientment gran, la $\mathbb{P}_{1/2}$-probabilitat que algun vèrtex de H_n estigui en un clúster obert infinit diguem que és com a mínim $1 - 10^{-6}$ (com H_n tendeix cap a T, aleshores la probabilitat que hi hagi un vèrtex en el clúster obert infinit és gairebé segur 1).

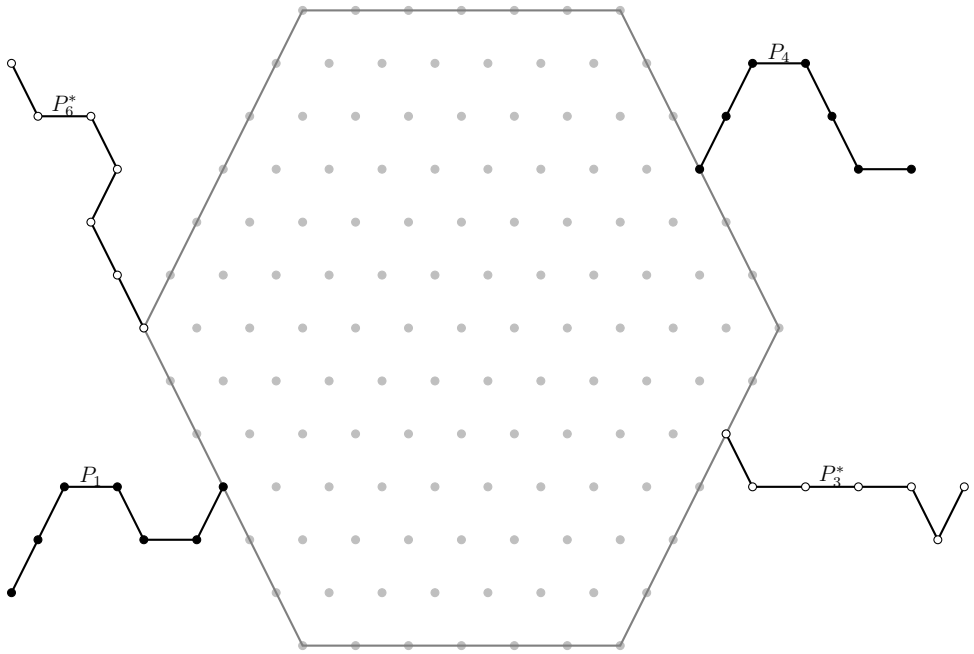

Figura 3.4: Representació de l'hexàgon H_n on cada costat conté n (en aquest cas $n = 7$) vèrtexs. Cada costat està enumerat de l'1 al 6, i els camins estan enumerats segons el costat del qual surten. Els camins infinits P_1 i P_4 són oberts i els camins infinits P_3^* i P_6^* són tancats

Enumerem cada costat de l'hexàgon amb un índex i, $i \in [1, 6]$ com en la Figura 3.4. Els vèrtexs que estan a les cantonades pertanyen als dos costats. Definim l'esdeveniment L_i com hi ha un camí obert infinit que surt de H_n pel costat i, i l'esdeveniment L_i^* com hi ha un camí tancat infinit que surt de H_n pel costat i. És a dir, l'esdeveniment L_i ens diu que hi ha un camí obert infinit en T amb un vèrtex inicial en el costat i de H_n i amb tots els altres vèrtexs fora de H_n; anàlogament l'esdeveniment L_i^* canviant obert per tancat. Per tant, $\bigcup_i L_i$ és l'esdeveniment que hi ha un clúster obert infinit que es troba amb un costat de H_n. En conseqüència, com un camí obert que comenci en un vèrtex de H_n ha de sortir per un dels costats de H_n, tenim $\mathbb{P}_{1/2}(\bigcup_i L_i) \geq 1 - 10^{-6}$. Els esdeveniments L_i són creixents i per tant L_i^C són decreixents. Per simetria tenim que $\mathbb{P}_{1/2}(L_i) = \mathbb{P}_{1/2}(L_1)$. Aleshores

$$\mathbb{P}_{1/2}(\bigcup_i L_i) = 1 - \mathbb{P}_{1/2}(\bigcup_i L_i)^C = 1 - \mathbb{P}_{1/2}(\bigcap_i L_i^C) \leq 1 - \mathbb{P}_{1/2}(L_i^C)^6$$

on hem usat el Lema de Harris i la simetria en l'última desigualtat. Per tant, com $\mathbb{P}_{1/2}(\bigcup_i L_i) \geq 1 - 10^{-6}$ tenim

$$\mathbb{P}_{1/2}(L_i^C) \leq 1/10$$

d'on es dedueix que

$$\mathbb{P}_{1/2}(L_i) = 1 - \mathbb{P}_{1/2}(L_i^C) \geq 1 - 1/10$$

Una aresta està tancada amb probabilitat $1-p$ i per tant $\mathbb{P}_p(L_i^*) = \mathbb{P}_{1-p}(L_i)$. Llavors $\mathbb{P}_{1/2}(L_i^*) = \mathbb{P}_{1/2}(L_i) \geq 1 - 1/10$.

Definim un esdeveniment $E = L_1 \cap L_3^* \cap L_4 \cap L_6^*$. Aquest esdeveniment E es compleix amb probabilitat com a mínim $6/10 > 0$, i és independent del valor dels vèrtexs de l'hexàgon H_{n-1}. Per tant, amb probabilitat > 0 l'esdeveniment E es compleix i cada vèrtex de H_{n-1} està tancat. Aleshores, els camins infinits tancats P_1^*, P_6^* que sabem que existeixen perquè L_3^*, L_6^* es compleixen, separen els camins oberts infinits P_1 i P_4, generats per L_1 i L_4. Aquests dos camins P_1 i P_4, com no es poden unir pel centre de H_{n-1} (tots els vèrtexs estan tancats) ni tampoc per fora (els camins tancats P_3^*, P_6^* són infinits), formen dos clústers oberts infinits, que contradiu el Teorema 3.6.

Per tant, podem concloure que $p_c^s(T) = 1/2$. □

3.3 Percolació en arestes en el reticle triangular

A continuació, donarem només una idea de com s'arriba a provar la probabilitat crítica per percolació en arestes en el reticle triangular. La idea principal sobre aquest resultat és el fet que les probabilitats crítiques per percolació en arestes en un reticle pla Λ i el seu dual Λ^* satisfan

$$p_c^b(\Lambda) + p_c^b(\Lambda^*) = 1 \tag{3.1}$$

Necessitem primer donar un nou reticle H, el reticle hexagonal. Aquest és simplement un conjunt d'hexàgons col·locats com un rusc d'abelles, on els vèrtexs són els vèrtexs d'aquests hexàgons, i les arestes els costats dels hexàgons. Es veu clarament que el reticle triangular T i l'hexagonal H són un dual de l'altre; veure figura 3.5. Per tant, tenim que

$$p_c^b(T) + p_c^b(H) = 1$$

El resultat que volem enunciar, però que no provarem és el següent.

Teorema 3.11. *Sigui T el reticle triangular i H el reticle hexagonal. Aleshores $p_c^b(T) = 2\sin(\pi/18)$ i $p_c^b(H) = 1 - 2\sin(\pi/18)$.*

Aquest teorema es prova usant el resultat (3.1) i un concepte anomenat transformació estrella-triangle que relaciona els reticles T i H, el qual explicarem a continuació.

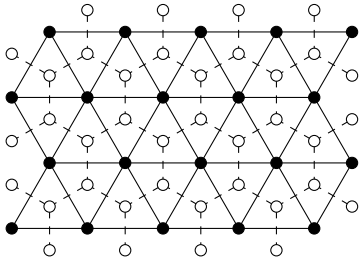

Figura 3.5: Les línies contínues formen el reticle triangular T, i les línies discontínues el reticle hexagonal H. Es pot observar aquests reticles són duals l'un de l'altre.

Siguin G_1 i G_2 dos grafs com en la Figura 3.6, on les arestes de G_1 són obertes amb probabilitat p_1 independentment, i les de G_2 són obertes amb probabilitat p_2 independentment. En els dos grafs, tenim 5 possibilitats de connectar els vèrtexs $\{x, y, z\}$ amb camins oberts: $\{\{x, y, z\}\}$, $\{\{x\}, \{y\}, \{z\}\}$, $\{\{x, y\}, \{z\}\}$, $\{\{x, z\}, \{y\}\}$, $\{\{y, z\}, \{x\}\}$, on dos vèrtexs estan connectats si es troben en el mateix conjunt, és a dir, dintre les claus. Aleshores, aquests casos es poden dividir en tres grups, i cada un d'ells té una certa probabilitat:

vèrtexs connectats	probabilitat en G_1	probabilitat en G_2
3	$p_1^3 + 3p_1^2(1-p_1)$	p_2^3
0	$(1-p_1)^3$	$(1-p_2)^3 + 3p_2(1-p_2)^2$
2	$p_1(1-p_1)^2$	$p_2^2(1-p_2)$

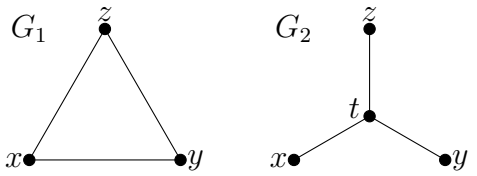

Figura 3.6: Representació dels grafs G_1 (triangle) i G_2 (estrella).

La taula ens dóna lloc a tres equacions

$$p_1^3 + 3p_1^2(1-p_1) = p_2^3$$

$$(1-p_1)^3 = (1-p_2)^3 + 3p_2(1-p_2)^2$$

$$p_1(1-p_1)^2 = p_2^2(1-p_2)$$

que resulten tenir solució. De l'última equació, deduïm que $p_2 = 1 - p_1$. Substituint això en la primera equació

$$p_1^3 + 3p_1^2(1-p_1) = (1-p_1)^3 \Leftrightarrow p_1^3 + 3p_1^2 - 3p_1^3 = -p_1^3 + 3p_1^2 - 3p_1 + 1 \Leftrightarrow$$

$$\Leftrightarrow p_1^3 - 3p_1 + 1 = 0$$

Aquesta equació té una única solució en $(0,1)$, $p_0 = 2\sin(\pi/18) = 0.3472...$

Per tant, usant el resultat (3.1) hauríem de veure que $p_c^b(T) = p_0$. Com ja hem dit abans, aquesta demostració és complicada i es pot trobar al llibre *Percolation* de Bollobás i Riordan [6]

Capítol 4

Estimant les probabilitats crítiques

El que hem fet en els apartats anteriors, bàsicament demostrar rigorosament alguna de les probabilitats crítiques en diversos reticles, està molt bé, però per mi seria impossible crear tots aquests resultats i demostracions des de zero. Així doncs, aquest capítol es tracta d'intentar (esperem aconseguir-ho) donar una estimació molt bona d'aquelles probabilitats crítiques, amb uns experiments semblants als realitzats per Hammersley l'any 1960. També aproximarem la probabilitat crítica del reticle $\mathbb{Z}^3$, encara que no hagi estat esmentat prèviament.

En tots els reticles que analitzem, donarem una estimació de la probabilitat crítica per percolació en arestes i vèrtexs. En tots els casos, la idea per estimar la probabilitat crítica, que és aquella que està relacionada amb l'existència o no d'un clúster obert infinit, és generar un reticle amb $n \times n$ punts ($n \times n \times n$ en el cas de $\mathbb{Z}^3$), i veure si existeix un camí que creua d'esquerra a dreta. Si $n \to \infty$ en principi la probabilitat crítica demostrada rigorosament i la que trobarem nosaltres hauria de ser la mateixa. Veiem doncs que procedim d'una manera similar a les demostracions d'aquestes probabilitats, ja que com hem vist en els capítols previs la idea era primer fer el cas finit i estendre-ho a l'infinit.

Per estimar aquesta probabilitat, la idea és, un cop tenim el reticle, anar obrint arestes (o vèrtexs) fins que aquest camí d'esquerra a dreta existeix, és a dir, percola. Un cop hem trobat aquest nombre d'arestes (o vèrtexs) necessàries perquè percoli, dividint-ho pel nombre total d'arestes (o vèrtexs) tenim una estimació de la probabilitat crítica. Aquest valor p que obtindrem, ens diu que si les arestes estan obertes amb probabilitat p o superior, aleshores aquest creuament d'esquerra a dreta existirà amb probabilitat més gran que zero (definició de probabilitat crítica). Per trobar una estimació

que sigui suficientment bona, repetim aquest procés unes quantes vegades. Hem realitzat dues estimacions per cada tipus de percolació. Pel reticle $\mathbb{Z}^2$ i el triangular, la primera estimació consisteix a calcular la probabilitat crítica en un reticle de mida 1000×1000, i repetir el procés 1000 vegades. En la segona, el reticle és de mida 10000×10000 i hem realitzat el procés 100 vegades. En el cas del reticle $\mathbb{Z}^3$ els experiments realitzats han estat amb reticles de mides $100 \times 100 \times 100$ i $500 \times 500 \times 500$ i repetint els processos 1000 i 100 vegades respectivament. L'ideal seria si poguéssim fer el reticle encara molt més gran, i repetir-ho moltes més vegades, però amb un ordinador normal el temps que arribaria a tardar és immens. Com veurem a continuació, el primer experiment realitzat dóna molts bons resultats per alguns reticles però no tan bons en alguns altres (fins i tot els podríem considerar erronis), tot i que el temps que tarda és molt poc. Per això vam decidir realitzar la segona estimació, que dóna una aproximació molt més bona en alguns d'aquests casos que la primera estimació fallava, però el temps que tarda també és molt més elevat.

Des del principi ja tenia clar que volia realitzar alguna cosa més pràctica dins el treball, i en buscar informació sobre algun cas més pràctic, vaig anar a parar en un vídeo, on es veia un programa que anava obrint vèrtexs i al final deia si percolava o no (entenent com a percolació el fet d'haver-hi un camí de dalt a baix). Vaig veure que aquest programa era la primera pràctica d'un curs en línia de *Coursera* (*Algorithms, part 1*), així que vaig decidir apuntar-me en aquest curs i realitzar-lo. La primera part del curs tractava de veure diversos algoritmes per connectar grafs, per tal de realitzar-ho de la manera més eficient. La pràctica que s'havia d'entregar eren dos programes: el primer consistia en donat una sèrie de vèrtexs en $\mathbb{Z}^2$, dir si existia un camí de dalt a baix, si percolava; el segon, es tractava de repetir aquest procés una certa quantitat de vegades per aproximar la probabilitat crítica de percolació en vèrtexs en aquest reticle. Els programes no estaven fets, però en el guió de la pràctica [17] apareixia les funcions bàsiques que havies d'implementar, així com un jar extern que et podies descarregar, sobretot útil en la cerca de camins entre vèrtexs d'un graf i tot el relacionat amb això. El principi vaig tenir certs problemes per realitzar la tasca, ja que s'havia de programar en Java, i no ho havia fet mai. Un cop passat aquesta primera fase (la d'aprendre les quatre coses bàsiques del Java) com durant la carrera havia programat amb c++ i com la implementació del programa no era res espectacular, tot va ser més fàcil.

Un cop vaig tenir la pràctica penjada, avaluada i superada (a més amb la màxima qualificació, no volia que le programa 'base' fallés), vaig començar a adaptar aquest a les meves necessitats. Per percolació en vèrtexs gairebé no vaig fer cap canvi, només el de buscar els camins d'esquerra a dreta en

comptes de dalt a baix, ja que durant tot el treball ho hem estat fent així. En els altres casos hi va haver una mica més de feina, però ja explicarem en cada subapartat com ho vam fer.

Aquest capítol constarà de sis parts, una per cada tipus de percolació, on explicarem com hem aconseguit els programes, com funcionen, i els resultats que n'hem obtingut. També hi ha un parell de vídeos per cada tipus de percolació (excepte pels casos de $\mathbb{Z}^3$) que mostra com es veu la interfície gràfica del programa. En principi, amb el *Flash Player* instal·lat i usant l'*Adobe* s'hauria de visualitzar en el mateix pdf. La qualitat no és la millor, ja que el fet d'haver de gravar primer la pantalla (el programa amb el qual ho he realitzat no ho gravava amb alta definició), després retocar el vídeo i finalment canviar-lo a format .swf ha fet que la qualitat anés baixant. Si no es pot visualitzar directament des del pdf, o es vol veure amb una mica més de qualitat, a cada descripció del vídeo hi ha un enllaç a *Youtube* on es pot veure.

4.1 Percolació en vèrtexs en $\mathbb{Z}^2$

Com hem explicat anteriorment, hem realitzat dos programes per cada situació, un que ens serveix per veure si donat un reticle $n \times n$ hi ha un camí d'esquerra a dreta, i l'altre que ens estima la probabilitat crítica.

La idea del primer programa consisteix a crear una matriu de booleans $n \cdot n$ que ens digui si el vèrtex i està obert o no, i afegir dos vèrtexs imaginaris, un a l'esquerra i un altre a la dreta del reticle, que seran els que ens indicaran si percola o no. És a dir, si existeix un camí obert que va del vèrtex que hem afegit a l'esquerra del reticle al vèrtex que hem afegit a la dreta, aleshores podem afirmar que percola. Aquest vèrtex nou de l'esquerra està connectat automàticament amb tots els vèrtexs de la primera columna, i el vèrtex que hem afegit a la dreta ho fa amb els de la n-èssima columna; veure Figura 4.1 . A l'hora d'unir dos vèrtexs, o veure si aquests dos estan connectats, usem un algorisme que estava en aquest jar extern que ens vam descarregar, anomenat *Weighted Quick Union*. Per això necessitem crear una classe d'aquest algoritme, consistent en els $n \cdot n$ vèrtexs del reticle, més els 2 que hem afegit a esquerra i dreta.

Aquest algorisme *Weighted Quick Union* és una modificació del *Quick Union*, que evita que hi hagi arbres molt alts i que per tant en el pitjor dels casos el temps que es tarda és menor. El gran avantatge del *Weighted Quick Union* és que uneix dos vèrtexs i diu si dos vèrtexs estan connectats amb cost $logN$, on N és el nombre total de vèrtex que hi ha (en el nostre cas $n \cdot n + 2$), a diferència del *Quick Union*, que ho feia amb cost N. La idea del *Quick*

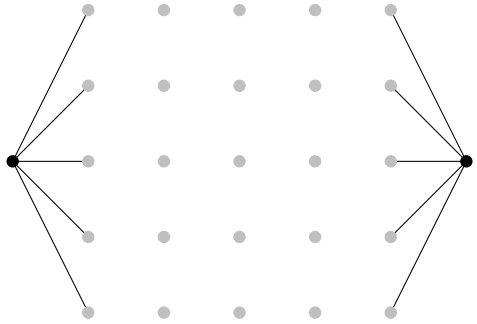

Figura 4.1: El reticle 5×5 amb els dos vèrtexs imaginaris a l'esquerra i dreta. Aquests vèrtexs estan connectats amb tots aquells situats a la dreta o l'esquerra d'ells. És a dir, el fet d'obrir un dels vèrtexs de la primera o última fila, automàticament implicarà la connexió amb els respectius vèrtexs imaginaris.

Union és crear un vector $id[]$ de pares dels vèrtexs, on l'arrel d'un vèrtex x s'expressa per $id[id[...id[x]...]]$. Per tant, un vèrtex és una arrel (està situat a la part superior de l'arbre) si el seu pare és ell mateix. Dos vèrtexs x i y estan connectats si les seves arrels són les mateixes. Per connectar dos vèrtexs x i y, canviem el pare de l'arrel de x per l'arrel de y. La diferència amb el *Weighted Quick Union* és que aquest, a l'hora d'unir dos vèrtexs, mira quin dels dos arbres té una alçada més gran i uneix el que té l'alçada més petita a l'alçada més gran. És a dir, en comptes d'unir sempre x a y i canviar el pare de l'arrel de x per l'arrel de y, a vegades uneix y amb x i és el pare de l'arrel de y el que canvia. Per veure més detalladament aquesta explicació sobre aquests dos algoritmes (potser aquesta explicació és una mica confusa) premeu aquí

El primer programa consta de 8 funcions diferents, algunes d'elles molt curtes (veure Annex 5.1):

1. La primera de totes consisteix en la creació dels vèrtexs del reticle $\mathbb{Z}^2$. Ho fem mitjançant una matriu de booleans, iniciats tots ells amb el valor *false*, ja que els vèrtexs inicialment estan tancats. El que és important recalcar és que cridem la classe *WeightedQuickUnionUF* dues vegades, una serà per veure si dos vèrtexs estan connectats, i l'altre per veure si a més d'estar connectats estan plens, és a dir, el camí que va d'esquerra a dreta passar per allà. Això és per evitar problemes a l'hora de crear la interfície gràfica: el fet de crear aquest nou vèrtex a la dreta, aquest pot estar connectat amb un seguit de vèrtexs que no es troben en el camí obert per on percola, i després el programa marcaria de color blau alguns vèrtexs que no formen part del camí obert com en

la Figura 4.2.

2. Aquesta funció simplement passa de les coordenades del vèrtex en el reticle (fila i columna, d'1 a n) a les coordenades en el vector *WeightedQuickUnion*, que van del 0 al $n \cdot\ n+1$, on el vèrtex de l'esquerra té associat el valor 0, i el de la dreta $n \cdot n + 1$.

3. Aquesta segurament és la funció més interessant. Es tracta de com el programa obre el vèrtex que li dius, i el que fa un cop l'has obert. Primer de tot, s'obre el vèrtex que hem cridat, és a dir, posa a *true* el seu valor en la matriu de booleans. Després considera els casos en els quals el vèrtex està situat a la primera o última columna. Si és així, com hem dit abans, el vèrtex es connecta automàticament amb el vèrtex 0 o el $n \cdot n + 1$ respectivament. A partir d'aquí, per cada vèrtex, mirem aquells vèrtexs que té a dalt, baix, esquerra i dreta (a no ser que no en tingui en una d'aquestes posicions). Si un (o els que siguin) d'aquests vèrtexs està obert, aleshores connecta aquests dos vèrtexs. En el programa podem observar que en tots els casos menys quan la columna és la n-èssima, fem dos tipus de connexions. Això és per evitar, com hem dit abans, un problema en la visualització gràfica.

4. El mateix nom indica què fa aquesta funció: donat un vèrtex en coordenades (i,j) diu si està obert o no.

5. Aquesta funció diu si un vèrtex està ple, és a dir, si el flux que es propaga des de l'esquerra arriba allà mitjançant vèrtexs oberts. És a dir, diu si el vèrtex, a més d'estar obert, pertany al camí obert que ha de creuar el rectangle. Només ens servirà per a la part gràfica.

6. Simplement compta el nombre de vèrtexs oberts que hi ha.

7. Diu si el reticle percola, és a dir, si hi ha aquest camí que va d'esquerra a dreta. Veure això és tan fàcil com veure si en la classe del *WeightedQuickUnionUF* hi ha un camí entre els dos vèrtexs imaginaris esquerre i dreta (aquí el motiu de l'agregació d'aquests dos vèrtexs).

8. És l'encarregada de generar la interfície gràfica. En aquest cas, ja ens la donàvem feta en el curs i no he hagut de tocar res. Tampoc no és difícil de comprendre. El que fa és dibuixar un quadrat negre deixant les vores de color blanc, i a partir d'aquí va afegir quadrats de color blanc (si el vèrtex està obert) o de color blau (si està obert i ple) a la posició corresponent. D'aquesta manera, les arestes que connectarien els vèrtexs no queden visibles, però es veu més clarament si un vèrtex

està obert o no. A la part inferior del gràfic hi apareix el nombre de vèrtexs oberts i si el sistema percola o no.

9. Finalment tenim el *main*, que ens servia per jugar una mica amb el programa i detectar errors mitjançant la interfície gràfica. Hem creat dues versions: la primera llegeix un fitxer i obra els vèrtexs que li has posat (molt útil per detectar errors) i la segona va obrint vèrtexs fins que el programa percola i dóna quina seria la probabilitat crítica (una estimació).

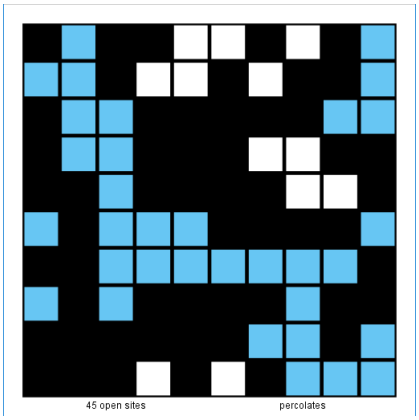

Figura 4.2: Si treballem només amb un vector *WeightedQuickUnionUF* pot passar que alguns vèrtexs es pintin equívocament. Per exemple, els quatre vèrtexs de dalt a la dreta no haurien d'estar pintats de color blau, ja que el camí obert que creua el rectangle no els conté.

El segon programa (veure Annex 5.1) consisteix a fer aquest procés que hem esmentat en la segona part del *main* tantes vegades com vulguem. Per generar la posició dels vèrtexs aleatòriament, generem aleatòriament dos nombres de l'1 a n. Després obrim el vèrtex corresponent i veiem si percola. Aquest procés d'obertura de vèrtexs es va realitzant fins que existeix aquest camí d'esquerra i dreta. Aleshores, ens guardem l'estimació de la probabilitat crítica, que no és res més que una divisió entre els vèrtexs oberts i el total de vèrtexs, i repetim aquest procés el nombre de vegades que prèviament havíem establert. Finalment, trobem una mitjana i desviació típica d'aquestes probabilitats crítiques, i per tant podem donar l'interval de confiança d'aquesta.

Executant aquest segon programa, amb un reticle de mida 1000×1000 i duen a terme el procés 1000 vegades obtenim els següents resultats:

Temps d'execució del programa	245889 mil·lisegons (uns 4 minuts)
Mitjana	0.5927396
Desviació típica	0.0030204
Interval confiança 95%	[0.5925523, 0.5929268]

Si ho fem amb un reticle 10000×10000 i 100 vegades obtenim:

Temps d'execució del programa	5235495 mil·lisegons (uns 87 minuts)
Mitjana	0.5927290
Desviació típica	5.4258449E-4
Interval confiança 95%	[0.5926226, 0.5928353]

Durant els apartats anteriors no havíem donat la probabilitat per percolació en vèrtexs en aquest reticle. Aquesta probabilitat no s'ha pogut aconseguir de manera teòrica, així que només se'n coneix una aproximació. Newmann i Ziff [6] van trobar una aproximació d'aquesta probabilitat crítica $p_c^s(\mathbb{Z}^2) = 0.59274621 \pm 0.00000013$ (pàg. 176). Jacobsen [18] va trobar una aproximació el 2015 consistent en 0.59274605079210. Com podem observar, la mitjana de les nostres dues aproximacions correspon amb els 4 primers decimals, el valor es troba dins l'interval de confiança i la desviació és molt petita. Per tant, els resultats obtinguts són molt satisfactoris. Entre les dues estimacions, els resultats no són gaire diferents: la segona estimació ens dóna un interval de confiança més petit, però la mitjana està una mica més lluny del valor 'exacte' que en la primera estimació. Amb el que sí que es diferencien, és amb el temps que ha tardat el programa a executar-se: passem de 4 minuts a 87!

Les Figures 4.3 i 4.4 mostren com és la interfície gràfica del programa, on es van obrint vèrtexs fins que el sistema percola.

4.2 Percolació en arestes en $\mathbb{Z}^2$

Per crear el programa que ens diu si donat un reticle amb unes certes arestes obertes existeix un camí d'esquerra a dreta, hem adaptat el programa que teníem fet per percolació en vèrtexs. En aquest cas, hem enumerat els vèrtexs i les arestes que els uneixen de la següent manera: els vèrtexs estan enumerats amb nombres successius de l'1 a n d'esquerra a dreta i de dalt a baix; les arestes estan enumerades primer les horitzontals (també d'esquerra a dreta i de dalt a baix) amb els nombres d'1 a $n^2 - n$ i després les verticals amb

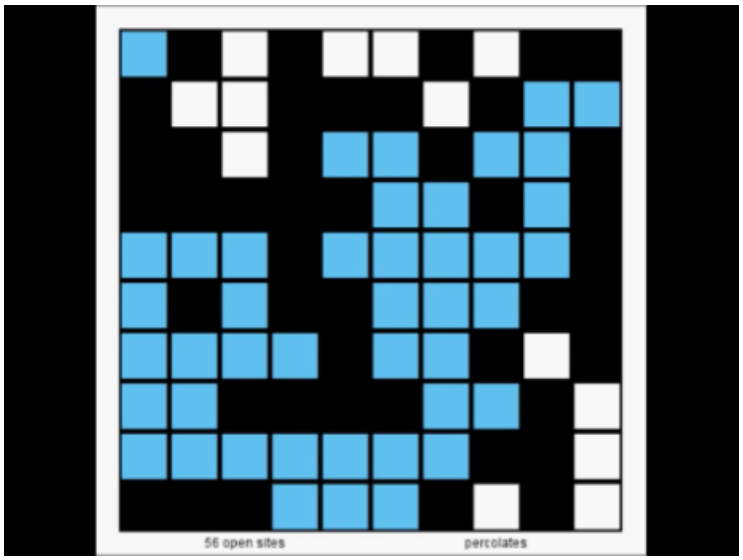

Figura 4.3: Percolació en vèrtexs en el reticle $\mathbb{Z}^2$ de mida 10×10. El color blau mostra els vèrtexs que formen o són candidats per formar part del creuament horitzontal d'esquerra a dreta. Link

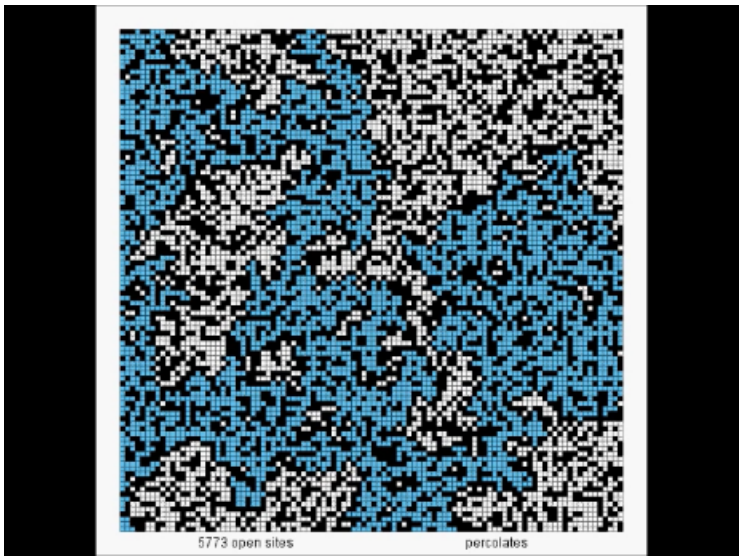

Figura 4.4: Percolació en vèrtexs en el reticle $\mathbb{Z}^2$ de mida 100×100. El color blau mostra els vèrtexs que formen o són candidats per formar part del creuament horitzontal d'esquerra a dreta. Link

els nombres de $n^2 - n + 1$ fins a $2(n^2 - n)$; veure Figura 4.5. Així doncs, sigui x una aresta horitzontal, aquesta uneix els vèrtexs y i $y + 1$, on $y = x + \lfloor (x-1)/(n-1) \rfloor$, on $\lfloor z \rfloor$ denota la part entera d'un nombre z. Si x és una aresta vertical, és a dir, té un valor entre $n^2 - n + 1$ i $2(n^2 - n)$, aleshores uneix els vèrtexs y i $y + n$, on $y = x\%(n^2 - n)$ amb l'excepció que $y = n^2 - n$ si $x\%(n^2 - n) = 0$. Aquestes relacions es troben fàcilment prenent un reticle relativament petit, veient que allà es compleix, i després mirant que també es satisfà en un altre reticle qualsevol. No és una prova rigorosa de què hagi de ser així, però ens serveix, i veient els resultats obtinguts sembla que funciona.

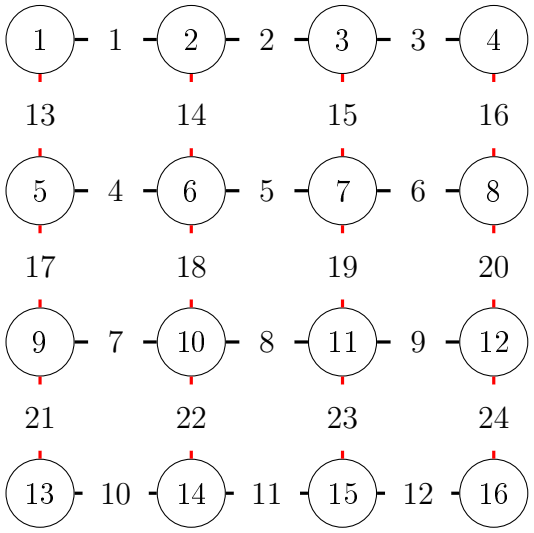

Figura 4.5: Cas $n = 4$. Els nodes numerats d'1 a $n \cdot n = 16$. Les arestes horitzontals (de color negre) enumerades d'1 a $n^2 - n = 12$ i les verticals (de color vermell) enumerades de $n^2 - n + 1 = 13$ a $2(n^2 - n) = 24$

Com abans, afegirem dos vèrtexs imaginaris que ens serviran per veure si percola o no, i els connectem amb els vèrtexs adjacents a aquests. També cridarem la classe *WeightedQuickUnion*, per facilitar la unió i busca de vèrtexs. Com observem en el primer programa (Annex 5.2), les funcions existents són més o menys les mateixes que en l'apartat anterior, i la funció que obra els vèrtexs fins i tot és més senzilla que abans (l'únic que s'ha de tenir en compte és els vèrtexs que connecta una aresta, que ho hem explicat abans). Hem afegit una funció nova *isFullBond)*, que és per veure si una aresta està plena i per tant l'hem de pintar. Per fer això mira si el vèrtex més petit dels dos que connecta l'aresta està ple (és a dir, està connectat amb l'esquerra) i l'aresta està oberta; si això passa, aleshores vol dir que l'aresta està plena. Potser una de la part més difícil d'aquest programa, va ser la de realitzar la interfície gràfica. Abans he dit que la part gràfica del programa de percolació en

vèrtexs en $\mathbb{Z}^2$ ja estava feta, i per tant no vaig haver de fer-hi res. El que es tractava en aquest cas era simplement dibuixar punts i arestes (a diferència dels quadrats que simbolitzaven els vèrtexs oberts en percolació en vèrtexs). Els punts, corresponents als vèrtexs, els podíem dibuixar al principi. Les arestes les anàvem dibuixant a mesura que les anàvem obrint, i les pintàvem de color blau si estaven connectades amb algun vèrtex de la part esquerra mitjançant un camí obert d'arestes. Per aconseguir dibuixar correctament les arestes, va ser bàsicament per prova i error. Sabia quins vèrtexs connectava cada aresta, però després havia de saber la posició dels vèrtexs en el reticle (aquesta posició estava desviada per un marge de 0,5 que havíem deixat a dalt i baix) i la direcció i llargada que havia de tenir l'aresta. La funció del programa és una mica embolicada, ja que havia d'anar restant i sumant diverses quantitats en la posició de l'inici i final de la línia corresponent a una aresta. Tot i això, com he comentat, per prova i error i amb una mica de paciència s'acaba aconseguint.

El segon programa, que ens estimarà la probabilitat crítica, és simplement una rèplica de l'anterior, canviant obrir vèrtexs aleatòriament per arestes, i canviant que cridi el programa de percolació en arestes en comptes del de vèrtexs; veure Annex 5.2.

A l'hora d'obtenir aquesta probabilitat crítica, hem simulat també un reticle de 1000×1000 1000 vegades. Els resultats obtinguts són els següents:

Temps d'execució del programa	269488 mil·lisegons (uns 4 minuts i mig)
Mitjana	0.5000936
Desviació típica	0.0024043
Interval confiança 95%	[0.4999445, 0.5002426]

Amb la simulació 10000×10000 100 vegades obtenim:

Temps d'execució del programa	5403483 mil·lisegons (uns 90 minuts)
Mitjana	0.5000249
Desviació típica	4.2954385E-4
Interval confiança 95%	[0.4999407, 0.5001091]

En el cas teòric, el que era el Teorema de Harris-Kesten, vam veure que la probabilitat crítica que hi hagi un clúster obert infinit, equivalent a l'existència d'aquest camí d'esquerra a dreta si $n \to \infty$, era de $1/2$. Observant els resultats obtinguts, podem veure que aquests són bonissims: la mitjana

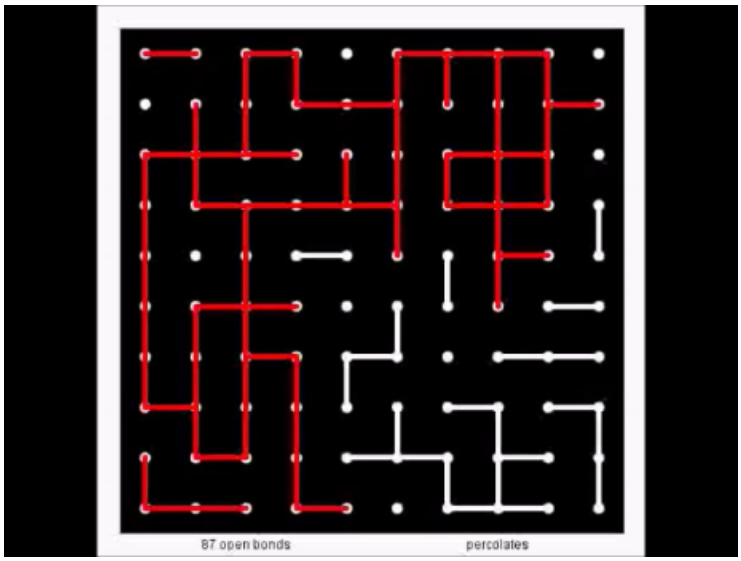

Figura 4.6: Percolació en arestes en el reticle $\mathbb{Z}^2$ de mida 10×10. El color vermell mostra les arestes que formen o són candidates per formar part del creuament horitzontal d'esquerra a dreta.Link

té fins a 4 decimals correctes, el valor $1/2$ es troba en l'interval de confiança i la desviació típica és molt petita. Veiem que, com abans, l'única diferència gran diferència entre les dues estimacions és la durada d'aquestes.

Aquest experiment seria semblant als experiments de Monte Carlo que es van realitzar abans de provar teòricament aquesta probabilitat crítica. Com llavors, els resultats ens fan pensar que efectivament la probabilitat ha de ser $1/2$. Observem que el temps que tarda en les dues estimacions és molt semblant al de l'apartat anterior. En els dos casos, el nombre d'elements a tenir en compte per obrir és de l'ordre de n^2: abans teníem n^2 vèrtexs possibles per obrir i ara $2(n^2 - n)$ arestes.

Les Figures 4.6 i 4.7 mostren com és la interfície gràfica del programa, on es van obrint arestes fins que el sistema percola.

4.3 Percolació en vèrtexs en el reticle triangular

A l'hora de realitzar els programes en el reticle triangular, teníem dues opcions. La primera era considerar el cas finit com un rombe, tal com està dibuixat en la Figura 3.3 del Lema 3.7. El problema era que si ens ho miràvem d'aquesta manera, a l'hora de dibuixar la interfície gràfica, el dibuix

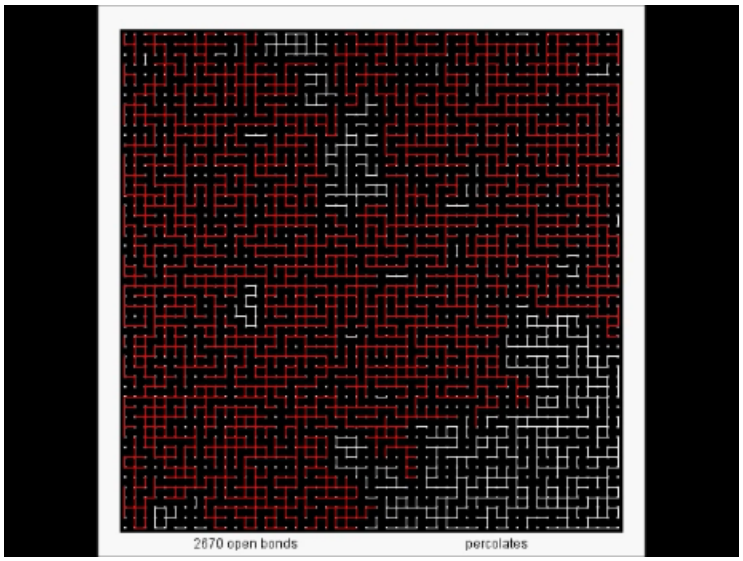

Figura 4.7: Percolació en arestes en el reticle $\mathbb{Z}^2$ de mida 50×50. El color vermell mostra les arestes que formen o són candidates per formar part del creuament horitzontal d'esquerra a dreta.Link

hagués quedat molt allargat i amb molts forats en buit si la mida del reticle era gaire gran. Per això vam decidir considerar-ho com en la Figura 3.1, és a dir, els vèrtexs com $\mathbb{Z}^2$ amb les files parelles (o imparelles) desplaçades horitzontalment per $1/2$, i amb les arestes que connecten aquells vèrtexs a distància 1.

En el lema 3.7 vam veure que hi ha una equivalència entre el reticle triangular i un compost per hexàgons: podem substituir cada vèrtex per un hexàgon, i existirà un camí d'esquerra a dreta en el reticle triangular si hi ha un conjunt d'hexàgons del mateix color que van d'esquerra a dreta. Així doncs, hem decidit que la interfície gràfica serà anar obrint aquests hexàgons, similarment com en percolació en vèrtexs en $\mathbb{Z}^2$, que obríem quadrats.

Pel que fa al primer programa (veure Annex 5.3), el que ens diu si hi ha percolació, és molt similar al de percolació en vèrtexs en $\mathbb{Z}^2$, amb la diferència que quan obrim un vèrtex s'han de mirar, a part dels vèrtexs situats a dalt i a baix, també els situats a les quatre diagonals; veure Figura 4.8. Les diagonals de color vermell són les arestes que abans eren verticals, i per tant ja teníem implementat el respectiu codi en el programa. Així doncs, només vam haver d'afegir les diagonals de color verd. Si la fila era parella (les files les comptem d'1 a n, de dalt a baix) miràvem les diagonals de l'esquerra, tenint en compte si estàvem a l'última fila o no, ja que aleshores no ens era necessari (de fet no podíem) mirar la diagonal inferior esquerra. Si la fila

era senar, miraven les diagonals de la dreta tenint en compte si la fila era la primera o no. L'altra gran diferència és a l'hora de crear i pintar la interfície gràfica, en aquest cas hexàgons. Sens dubte ha estat el gràfic més costós de realitzar dels quatre programes pels quals hem creat una part gràfica. Per fer-lo em vaig basar, per descomptat, en el gràfic creat en el primer programa de tots. Allà teníem l'espai dividit en quadrats 1×1, que ocupaven tot aquest espai. Les files parelles tenen, com en aquell cas, el centre de l'hexàgon en el centre del quadrat; les files senars, en canvi, tenen aquest centre desplaçat per 1/2. Un cop considerat això, tenint en compte que els hexàgons han de tenir tots els costats iguals i com han d'estar encaixats aquests hexàgons, usant trigonometria bàsica s'obté com s'han de dibuixar. Com abans, per prova i error i amb una mica de paciència, obtenim per cada vèrtex la posició dels 6 vèrtexs que formen els hexàgons. Observant els vídeos es pot veure que hi ha algun petit marge de color negre, per la forma com estan distribuïts els hexàgons.

El segon programa, per obtenir l'estimació de la probabilitat crítica, és idèntic al de percolació en vèrtexs en $\mathbb{Z}^2$ (canviant que crida la funció que diu si percola per vèrtexs en el reticle triangular), i fent una simulació 1000×1000 1000 vegades obtenim:

Temps d'execució del programa	180699 mil·lisegons (uns 3 minuts)
Mitjana	0.5005367
Desviació típica	0.0029800
Interval confiança 95%	[0.5003520, 0.5007214]

Fent la simulació amb un reticle 10000×10000 100 vegades:

Temps d'execució del programa	5205657 mil·lisegons (uns 87 minuts)
Mitjana	0.5000660
Desviació típica	5.6119011E-4
Interval confiança 95%	[0.4999560, 0.5001760]

En el capítol 3 hem vist que aquesta probabilitat crítica era 1/2. Els resultats de la primera simulació podrien semblar bons; mitjana molt propera a 1/2, desviació típica petita... Però si ens fixem en l'interval de confiança del 95%, veiem que el valor 1/2 no hi pertany! Primer de tot vaig executar unes quantes vegades el programa amb el reticle 1000×1000 1000 vegades (a no ser que hagués tingut moltíssima mala sort en el primer intent) però el valor 1/2 continuava sense estar dins l'interval de confiança. Això em va fer

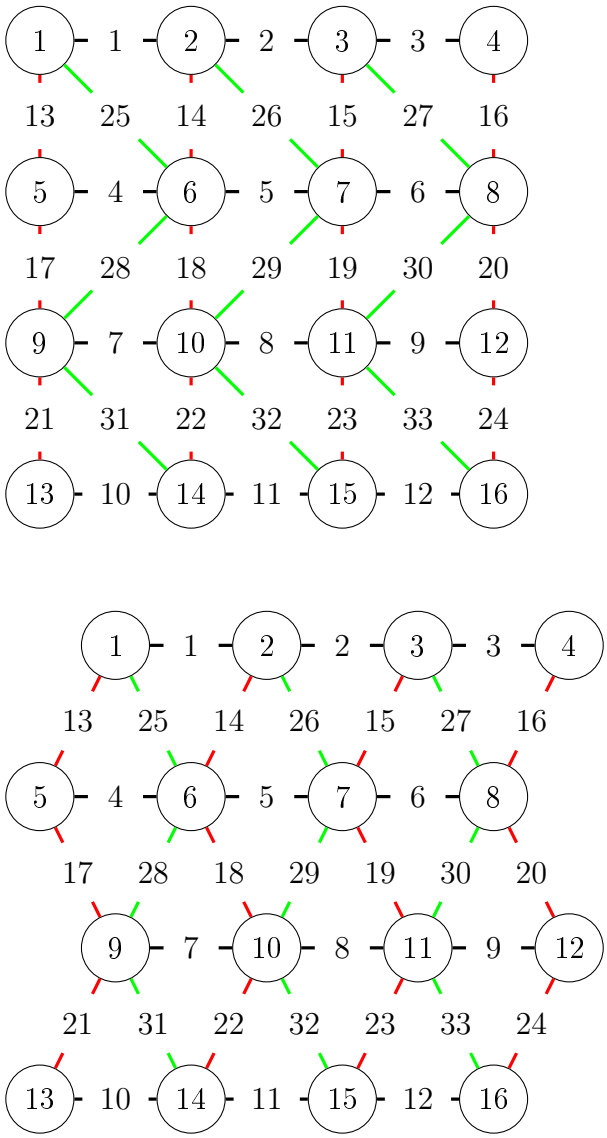

Figura 4.8: Representació del reticle triangular per $n = 4$. Si ens hi fixem, no és res més que la representació del reticle $\mathbb{Z}^2$ desplaçant les files parelles de vèrtexs i afegint les arestes verdes. Les arestes negres i vermelles estan enumerades com abans, les verdes de $2(n^2 - n) + 1 = 25$ fins a $2(n^2 - n) + (n - 1)^2 = 33$. En el gràfic de sota, les diagonals vermelles són les que anteriorment eren les arestes verticals.

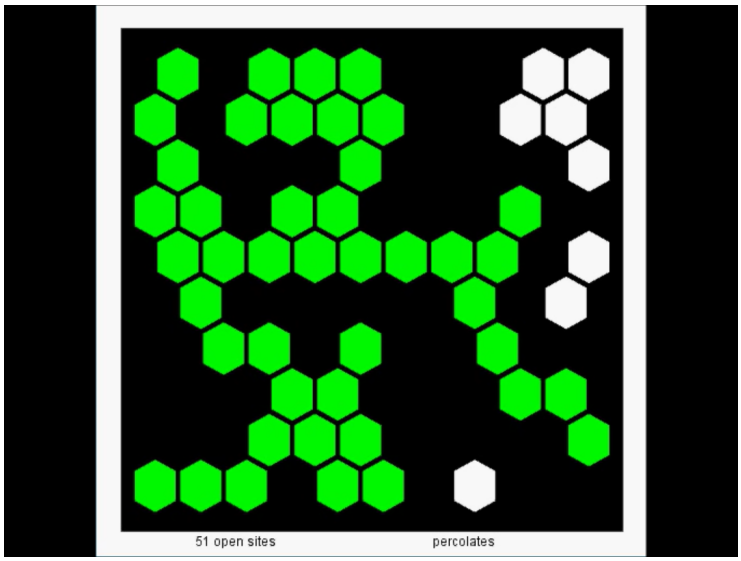

Figura 4.9: Percolació en vèrtexs en el reticle triangular de mida 10×10. El color verd mostra els vèrtexs que formen o són candidats per formar part del creuament horitzontal d'esquerra a dreta.Link

pensar que potser havia fet alguna cosa malament en el programa: el fet de no considerar alguna aresta, o fer un petit error podria portar-me a aquesta conseqüència. Així doncs, vaig decidir fer una simulació per un reticle més gran (no la vaig poder realitzar 1000 vegades, ja que era massa temps) i vaig obtenir uns resultats molt satisfactoris. En aquest cas, a més de tenir una mitjana molt bona i una desviació típica encara més petita, el valor $1/2$ ja estava en l'interval de confiança.

Les Figures 4.9 i 4.10 mostren com és la interfície gràfica del programa, on es van obrint vèrtexs fins que el sistema percola.

4.4 Percolació en arestes en el reticle triangular

Per realitzar aquest programa, la clau està en observar les similituds entre el reticle $\mathbb{Z}^2$ i el triangular. Com ja hem vist en el cas anterior i en la Figura 4.8, el reticle triangular és una simple ampliació (augmentem el nombre d'arestes) del reticle $\mathbb{Z}^2$. Observant aquesta Figura 4.8, veiem que eliminant les arestes de color verd tenim el mateix nombre d'arestes que en el reticle $\mathbb{Z}^2$, i les podem enumerar de la mateixa manera. En aquest cas, les arestes verticals estan en diagonal, però és pel fet que les files parelles (o imparelles) estan desplaçades $1/2$. Així doncs, enumerem les arestes negres i vermelles de

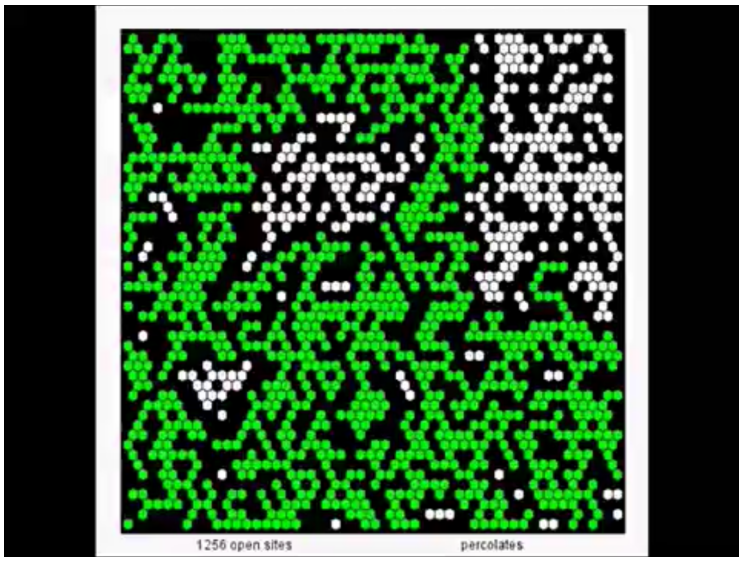

Figura 4.10: Percolació en vèrtexs en el reticle triangular de mida 50×50. El color verd mostra els vèrtexs que formen o són candidats per formar part del creuament horitzontal d'esquerra a dreta.Link

la mateixa manera que en el segon subapartat d'aquest capítol, de l'1 a $n^2 - n$ per les horitzontals i del $n^2 - n + 1$ a $2(n^2 - n)$ per les verticals (diagonals de color vermell). Els vèrtexs que connecten aquestes arestes són els mateixos que connectaven en $\mathbb{Z}^2$. Així doncs, l'únic que ens queda és afegir aquestes arestes en diagonal, de color verd en la Figura 4.8. Aquestes estaran enumerades de $2(n^2 - n) + 1$ a $2(n^2 - n) + (n - 1)^2$, el nombre total d'arestes en aquest reticle. També seguiran l'ordre de sempre: d'esquerra a dreta i de dalt a baix. En aquest cas, per cada fila no hi ha tantes arestes com vèrtexs, i per tant no podem fer simplement el mòdul per $n^2 - n$ com per les arestes vermelles i dir que uneixen el vèrtex obtingut amb el mòdul, i aquest sumant-li n. Aquí tenim una aresta menys per cada fila de vèrtex. Per tant, aquestes arestes que falten les calcularem amb l'expressió $\lfloor (x-1)/(n-1) \rfloor$ (de fet és el nombre d'arestes que falten menys una) on x és el valor de l'aresta mòdul $n^2 - n$. Després, depenent si el vèrtex es troba en una fila parella o imparella (si el valor que hem calculat és imparell o parell), obtenim els vèrtexs que uneix l'aresta. Com en els altres casos, per obtenir aquests vèrtexs que uneix l'aresta, ho he fet en un cas amb n relativament petit, i he mirat per si un n més gran també funcionava. L'expressió pot semblar una mica confusa, però és relativament fàcil de trobar.

Un cop tenim aclarida aquesta equivalència, el primer programa surt relativament fàcil simplement afegint aquestes noves arestes; veure Annex 5.4.

Potser la part més delicada és la interfície gràfica i la funció necessària per a aquesta, *isFullBond*. Amb calma, anar provant el programa i veient el què dibuixa, s'acaben trobant els algoritmes necessaris. El segon programa és anàleg als altres, cridant el procés de percolació que necessitem en aquest cas.

Executant el segon programa per un reticle 1000×1000 i 1000 vegades obtenim els resultats:

Temps d'execució del programa	341515 mil·lisegons (uns 5 minuts i mig)
Mitjana	0.3475547
Desviació típica	0.0018983
Interval confiança 95%	[0.3474370, 0.3476724]

En el cas en què el reticle és 10000×10000 i el procés es repeteix 100 vegades:

Temps d'execució del programa	6390574 mil·lisegons (uns 106 minuts)
Mitjana	0.3473366
Desviació típica	3.2354271E-4
Interval confiança 95%	[0.3472732, 0.3474001]

En el capítol 3 hem donat les indicacions i teoremes que porten a demostrar rigorosament que la probabilitat crítica en percolació en arestes en el reticle triangular és $2\sin(\pi/18) = 0.347296355....$ Com en el cas anterior, la simulació per $n = 1000$ ens dóna un interval de confiança on el valor $2\sin(\pi/18)$ no hi està inclòs (tot i donar-nos una bona mitjana). En canvi, la simulació per $n = 10000$ ja conté el valor. La mitjana i desviació típica d'aquesta segona simulació són molt bones. Per tant, un cop més, obtenim una aproximació molt bona del valor real.

Les Figures 4.11 i 4.12 mostren com és la interfície gràfica del programa, on es van obrint arestes fins que el sistema percola.

4.5 Percolació en altres reticles

És cert que les estimacions que hem fet en els capítols previs podrien semblar innecessàries, ja que s'ha demostrat teòricament el resultat d'aquestes (excepte el cas de percolació en vèrtexs en $\mathbb{Z}^2$). Tot i això he volgut fer aquestes estimacions, ja que és la manera natural de procedir quan tractes amb altres

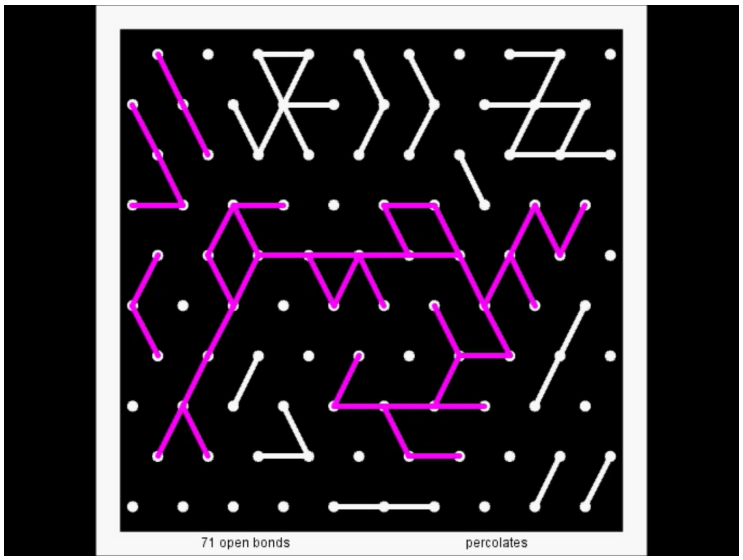

Figura 4.11: Percolació en arestes en el reticle triangular de mida 10×10. El color magenta mostra les arestes que formen o són candidates per formar part del creuament horitzontal d'esquerra a dreta.Link

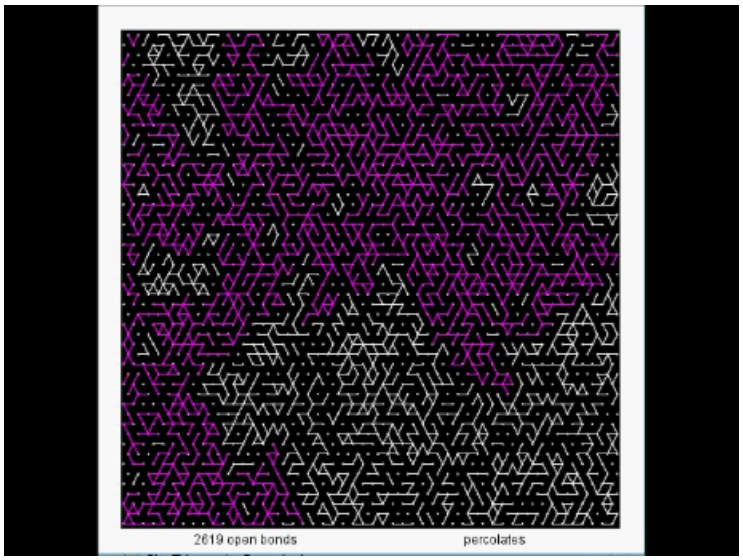

Figura 4.12: Percolació en arestes en el reticle triangular de mida 50×50. El color magenta mostra les arestes que formen o són candidates per formar part del creuament horitzontal d'esquerra a dreta.Link

reticles, que no són tan fàcils com $\mathbb{Z}^2$ o el reticle triangular. D'aquests dos se'n coneixen moltes propietats (n'hem vist algunes de les més importants durant el treball), però si sortim d'aquí els resultats coneguts són ben pocs. En aquest apartat, hem volgut estimar les probabilitats crítiques d'un reticle del qual no se'n coneixen rigorosament tantes propietats, el reticle cúbic o $\mathbb{Z}^3$. Després comprovarem si les estimacions que ens donen es corresponen amb les trobades prèviament per altres matemàtics amb eines molt més potents.

4.5.1 Percolació en vèrtexs en $\mathbb{Z}^3$

Hem creat un programa que ens estima la probabilitat crítica en percolació en vèrtexs en $\mathbb{Z}^3$; veure Annex 5.5. En aquest cas no he realitzat la interfície gràfica, ja que em va semblar massa difícil de fer i tampoc tenia molt temps per dedicar-m'hi. El programa ha set relativament fàcil de fer, adaptant el que ja teníem per percolació en vèrtexs $\mathbb{Z}^2$ i tenint en compte una observació clau. La diferència entre 2D i 3D, és l'aparició de la profunditat. Per tant, en comptes de tenir els vèrtexs precisats per dos índexs, ara en tindrem tres. Per tant, en lloc de tenir n^2 vèrtexs en tindrem n^3. Així doncs, el que hem de canviar del programa en $\mathbb{Z}^2$ és el nombre de vèrtexs, la matriu de booleans (que passarà a tenir tres components de mida n cadascuna), la funció que relaciona una tripleta (x,y,z) amb el seu nombre corresponent dins els n^3 vèrtexs i la funció que obre un vèrtex. Després, haurem d'adaptar les altres funcions, tot i que en la majoria de casos és només afegir la profunditat. A més, com no realitzem la part gràfica, hi ha alguna cosa que ens podem estalviar, com la funció per veure si un vèrtex està ple.

La funció *índex* s'adapta fàcilment de $\mathbb{Z}^2$ a $\mathbb{Z}^3$ si ordenem els vèrtexs de dalt a baix i d'esquerra a dreta com en $\mathbb{Z}^2$ i de dintre cap a fora en les diferents cares. És a dir, si té profunditat 1 no hem de sumar res a la funció índex que teníem en $\mathbb{Z}^2$, si té profunditat 2 hem de sumar n^2, si té profunditat 3 hem de sumar $2n^2$,..., si té profunditat n hem de sumar $(n-1)n^2$.

Per adaptar la funció que obre un vèrtex qualsevol, l'únic que hem d'afegir (a part de passar-ho tot de 2D a 3D, és a dir, afegint la variable de profunditat) és que miri si un vèrtex el podem connectar amb el que té, en la mateixa posició (fila i columna), en les cares de davant i darrere, sempre que sigui possible.

Un cop hem fet aquests petits canvis, ja podem aproximar la probabilitat crítica. En aquest cas, no prendrem $n = 1000$, ja que com estem en $\mathbb{Z}^3$ en comptes de $\mathbb{Z}^2$ el nombre de vèrtexs seria de mil milions en lloc del milió que teníem anteriorment. Aquest nombre, a més de ser una barbaritat, ens dóna un error de memòria quan executem el programa. Prendrem $n = 100$, que si ens hi fixem seran els mateixos vèrtexs que teníem per $\mathbb{Z}^2$ amb $n = 1000$.

Executant el programa també 1000 vegades, obtenim els següents resultats:

Temps d'execució del programa	207272 mil·lisegons (uns 3 minuts i mig)
Mitjana	0.3132171
Desviació típica	0.0030545
Interval confiança 95%	[0.3130278, 0.3134065]

La segona estimació la farem prenent $n = 500$. En aquest cas tindrem 125 milions de vèrtexs, molt semblant als 100 milions que teníem pels reticles en el pla. Executant també el procés 100 veegades obtenim:

Temps d'execució del programa	4099512 milisegons (uns 68 minuts)
Mitjana	0.3117790
Desviació típica	5.2306097E-4
Interval confiança 95%	[0.3116765, 0.3118816]

L'any 2014 Xiao, Wang, Lv i Deng [19] van trobar que una aproximació d'aquesta probabilitat era 0.31160768. Observant les nostres simulacions veiem que la primera dóna una aproximació acceptable, i la segona ja és molt bona. Tot i això, en els dos casos el valor exacte no pertany a l'interval de confiança: en la primera simulació està una mica lluny, mentre que en la segona no està dins per molt poc. Tanmateix, no crec que els programes que he realitzat estiguin malament, sinó que necessitaria executar el programa per una mida n més gran per obtenir-ne uns millors resultats. Tenint en compte tot això, també quedo satisfet amb l'aproximació que dóna aquest programa.

4.5.2 Percolació en arestes en $\mathbb{Z}^3$

Finalment, hem fet un programa per estimar la probabilitat crítica per percolació en arestes en $\mathbb{Z}^3$; veure Annex 5.6. Aquest ens ha costat bastant més que l'anterior, i també l'hem obtingut adaptant el programa que teníem, en aquest cas, per percolació en arestes en $\mathbb{Z}^2$. La idea ha set mirar com seria pel cas $n = 3$, i estendre-ho per a qualsevol n. Primer de tot (i com vam fer per 2D), hem triat com enumeraríem les arestes. Hem decidit enumerar primer les arestes de les cares, de la primera cara a l'última, i en cada cara les arestes estan enumerades primer les horitzontals (d'esquerra a dreta i de dalt a baix) i després les verticals (també d'esquerra a dreta i de dalt a baix), com en $\mathbb{Z}^2$. Les arestes que queden per enumerar són, doncs, aquelles que

connecten les cares. Estan enumerades també d'esquerra a dreta i de dalt a baix, començant per la cara més propera i anar tirant enrere. La Figura 4.13 ens ajuda a entendre el que acabo d'explicar.

Un cop sabia com estaven enumerades les arestes, he hagut de modificar el programa de $\mathbb{Z}^2$. Com abans, la part gràfica i les funcions necessàries per a aquesta les he pogut eliminar. He hagut de fer, primer de tot, alguns petits canvis bàsicament relacionats amb augmentar el nombre d'arestes i vèrtexs. La part més delicada ha estat la funció per obrir una nova aresta, i veure quins vèrtexs connecta. Primer de tot, he separat les arestes en dos casos, un si l'aresta pertany en una de les cares, i l'altre si l'aresta uneix aquestes cares. Després havia de detectar en quina de les cares es trobava l'aresta, i si era vertical o horitzontal. Per fer-ho, hem dividit aquestes $2(n^3 - n^2)$ arestes pel nombre de cares que hi havia, usant aquesta funció '*for*', amb un índex que va avançant d'1 a $2n$ de dos en dos. Per cada cara, hem mirat si l'aresta era vertical o horitzontal com en $\mathbb{Z}^2$, mirant si es trobava en les $n^2 - n$ primeres arestes de la seva cara, o estava entre la $n^2 - n + 1$ i la $2(n^2 - n)$. Un cop tenia això, feia el mateix que en el programa per $\mathbb{Z}^2$, sumant n^2 als vèrtexs que connectava l'aresta tantes vegades com la cara que fos (he considerat en aquest cas que la primera cara és la 0, i l'última la (n-1)-èssima). Per les arestes que connecten les cares, simplement es tracta de calcular el nombre $z = x\%(n^3 - n^2)$, on x és el valor de l'aresta. Després aquesta aresta x es veu fàcilment que connecta els vèrtexs z i $z + n^2$.

Com en el cas anterior, per calcular la probabilitat crítica, hem fet dues simulacions. Per $n = 100$ i realitzant el procés 1000 vegades, els resultats obtinguts són els següents:

Temps d'execució del programa	277761 mil·lisegons (uns 4 minuts i mig)
Mitjana	0.2498065
Desviació típica	0.0018413
Interval confiança 95%	[0.2496924, 0.2499206]

Per $n = 500$ i 100 vegades:

Temps d'execució del programa	5139379 mil·lisegons (uns 85 minuts)
Mitjana	0.2489113
Desviació típica	3.0070512E-4
Interval confiança 95%	[0.2488523, 0.2489702]

L'any 2013 Wang, Zhou, Zhang, Garoni i Deng [20] van aproximar aquesta probabilitat per 0.24881182. Com abans la segona estimació s'hi acosta molt,

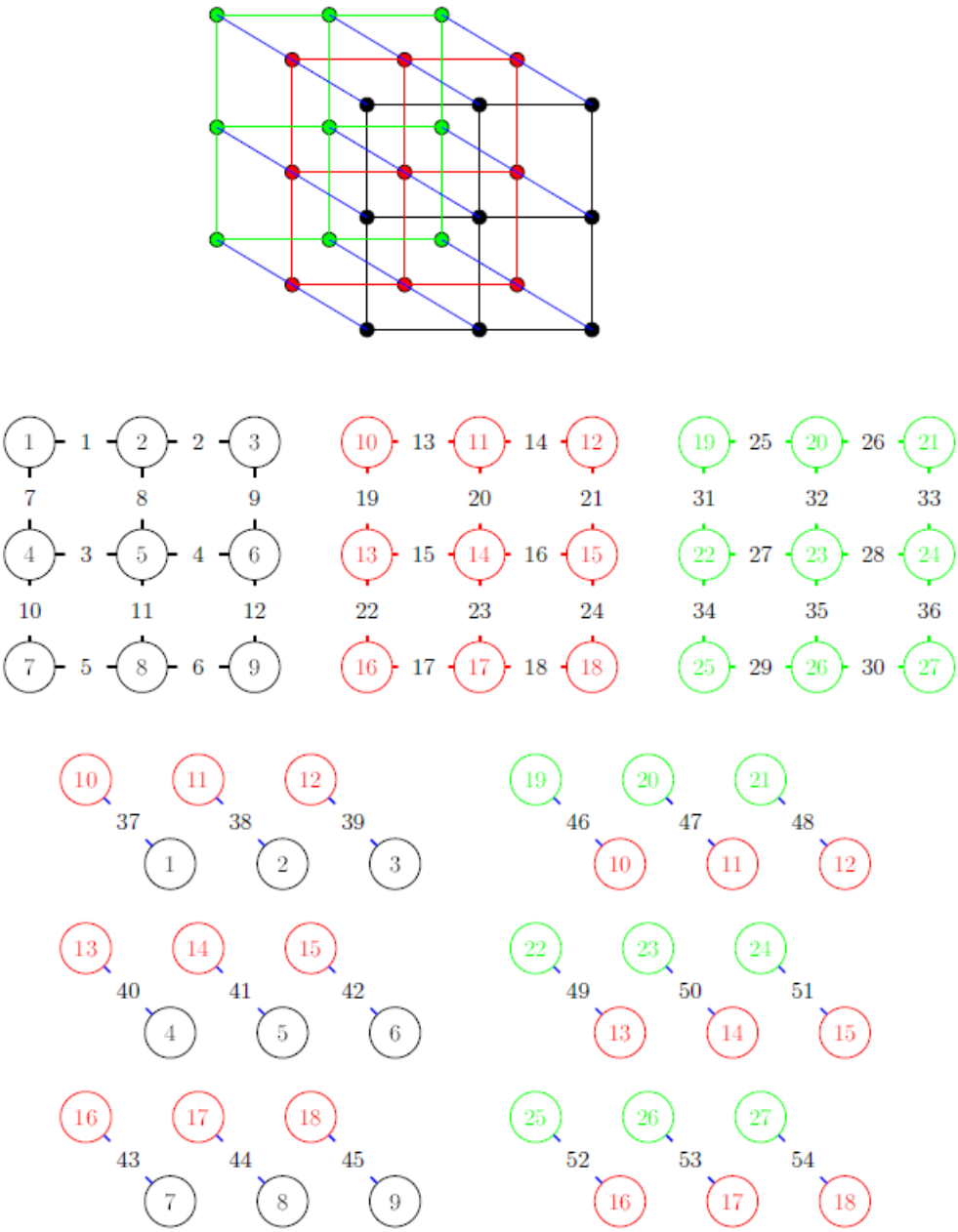

Figura 4.13: Representació d'un cub per $n = 3$. Enumerem els vèrtexs d'esquerra a dreta i de dalt a baix per cada cara, i de la primera cara a l'última. Les arestes estan numerades primer aquelles contingudes en cada cara com en $\mathbb{Z}^2$ i després aquelles que connecten les cares, d'esquerra a dreta i de dalt a baix, i de la primera cara a l'última

encara que aquest valor no es trobi en l'interval de confiança. També ha de ser degut al fet que la n que considerem és força petita, és només 500. Estic convençut que amb una n major l'aproximació seria encara millor.

Capítol 5

Annex

5.1 Programes percolació en vèrtexs en Z^2

Primer programa:

```
import java.awt.Font;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

import edu.princeton.cs.algs4.StdDraw;
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class SitePercolation {

private int size;
private int left = 0;
private int right;
private WeightedQuickUnionUF wquf; // ens serveix per pintar el cami de color
private WeightedQuickUnionUF wqufPerc; // aquesta ens servira per no confodre connectat amb ple
private boolean [][] opened;

public SitePercolation(int n) {          // creem el reticle nxn, amb tots els vrtexs tancats
if (n <= 0) throw new IllegalArgumentException("Given n <= 0");
size = n;
right = size*size + 1;
wquf = new WeightedQuickUnionUF(size*size + 2); // +2 pels dos vrtexs que afegim, dreta i esquerra
wqufPerc = new WeightedQuickUnionUF(size*size + 2);
opened = new boolean[size][size];
}


private int index(int row, int col) {   // serveix per passar de coordenades (row,col) a la posici del vector
return size*(row-1) + col;
}


public void open(int row, int col) {
opened[row-1][col-1] = true;
if (col == 1) {
wquf.union(index(row,col), left);
wqufPerc.union(index(row,col), left);
}
if (col == size) {
wqufPerc.union(index(row,col), right);
}
if (col > 1 && isOpen(row, col-1)) {        //mirem si podem connectar amb vrtex de l'esquerra
wquf.union(index(row, col-1), index(row, col));
wqufPerc.union(index(row, col-1), index(row, col));
}
if (col < size && isOpen(row, col+1)) {     //mirem si podem connectar amb vrtex de la dreta
```

```
wquf.union(index(row, col+1), index(row, col));
wqufPerc.union(index(row, col+1), index(row, col));
}
if (row > 1 && isOpen(row-1, col)) {    //mirem si podem connectar amb vrtex de sobre
wquf.union(index(row-1, col), index(row, col));
wqufPerc.union(index(row-1, col), index(row, col));
}
if (row < size && isOpen(row+1, col)) {     //mirem si podem connectar amb vrtex de sota
wquf.union(index(row+1, col), index(row, col));
wqufPerc.union(index(row+1, col), index(row, col));
}
}


public boolean isOpen(int row, int col) {
return opened[row-1][col-1];
}


public boolean isFull(int row, int col) {   // serveix per la part  grfica
if (row > 0 && row <= size && col > 0 && col <= size) {
return wquf.connected(left, index(row,col));
}
else {
throw new IndexOutOfBoundsException();
}
}


public int numberOfOpenSites() {
int comptador = 0;
for (int i = 1; i <= size; ++i) {
for (int j = 1; j <= size; ++j) {
if (isOpen(i,j)) ++comptador;
}
}
return comptador;
}


public boolean percolates() {
return wqufPerc.connected(left, right);
}


// Visualitzador
private static final int DELAY = 1;
private static Scanner in;

public static void draw(SitePercolation perc, int n) {
StdDraw.clear();
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.setXscale(-0.05*n, 1.05*n);
StdDraw.setYscale(-0.05*n, 1.05*n);  // deixem marge per escriure text
StdDraw.filledSquare(n/2.0, n/2.0, n/2.0);


int opened = 0;
for (int row = 1; row <= n; row++) {
for (int col = 1; col <= n; col++) {
if (perc.isFull(row, col)) {
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
opened++;
}
else if (perc.isOpen(row, col)) {
StdDraw.setPenColor(StdDraw.WHITE);
opened++;
}
else
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.filledSquare(col - 0.5, n - row + 0.5, 0.45);
}
}

// Escrivim el nombre de vrtexs oberts i si percola o no a la part inferior del  grfic
StdDraw.setFont(new Font("SansSerif", Font.PLAIN, 12));
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.text(0.25*n, -0.025*n, opened + " open sites");
if (perc.percolates()) StdDraw.text(0.75*n, -0.025*n, "percolates");
else                   StdDraw.text(0.75*n, -0.025*n, "does not percolate");

}
```

```
public static void main(String[] args) throws FileNotFoundException {

//main que obra un fitxer
/*

BufferedReader fitxer = new BufferedReader(new FileReader
("C:\\Users\\PC\\workspace\\Percolation_and_visualizer\\Exemples\\jerry47.txt"));

in = new Scanner(fitxer);
int size = in.nextInt();
StdDraw.enableDoubleBuffering(); // engegem l'animaci
Percolation perc = new Percolation(size);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);

while (in.hasNextInt()) {
int row = in.nextInt();
int col = in.nextInt();
perc.open(row, col);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);
}

if (perc.percolates()) StdOut.println("percola");
else StdOut.println("no percola");
*/


//main que obra vrtexs fins que percola

int size = StdIn.readInt();
StdDraw.enableDoubleBuffering();
SitePercolation perc = new SitePercolation(size);
draw(perc, size);
StdDraw.show();

while (!perc.percolates()) {
int row = StdRandom.uniform(size)+1;
int col = StdRandom.uniform(size)+1;
if (!perc.isOpen(row,col)) {
perc.open(row,col);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);
}
}

StdOut.println("probabilitat critica = " + (double) perc.numberOfOpenSites() / (double) (size*size));

}
}
```

Segon programa:

```
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.StdStats;

import java.text.DecimalFormat;

public class MonteCarlo {

private int size;
private int trials;
private double[] criticprob;

public MonteCarlo(int n, int t) {    // executa t vegades el programa de percolaci en vrtexs per un reticle nxn
if (n <= 0 || t <= 0) {
throw new IllegalArgumentException("Given n <= 0 || t <= 0");
}
size = n;
trials = t;
criticprob = new double[trials];
for (int i = 0; i < trials; ++i) {
SitePercolation perc = new SitePercolation(size);
int comptador = 0;
```

```
while (!perc.percolates()) {
int row = StdRandom.uniform(size)+1;
int col = StdRandom.uniform(size)+1;
if (!perc.isOpen(row, col)) {
perc.open(row, col);
comptador++;
}
}
criticprob[i] = (double) comptador/(double) (size*size);
StdOut.println("fi de la vegada " + i);
}
}


public double mean() {
return StdStats.mean(criticprob);
}


public double stddev() {
return StdStats.stddev(criticprob);
}


public double confidenceLo() {
return mean()-(1.96*stddev())/Math.sqrt(trials);
}


public double confidenceHi() {
return mean()+(1.96*stddev())/Math.sqrt(trials);
}



public static void main(String[] args) {
long time_start, time_end; // serveixen per calcular el temps que tarda el programa
int n = StdIn.readInt();
int T = StdIn.readInt();
time_start = System.currentTimeMillis();
MonteCarlo monte = new MonteCarlo(n, T);
time_end = System.currentTimeMillis();
StdOut.println("L'execucio del programa ha durat "+ ( time_end - time_start ) +" milisegons");
StdOut.println("Mitjana                = " + monte.mean());
StdOut.println("Desviacio tipica       = " + monte.stddev());
StdOut.println("Interval confianca 95% = [" + monte.confidenceLo() + ", " + monte.confidenceHi() + "]");
}

}
```

5.2 Programes percolació en arestes en $\mathbb{Z}^2$

Primer programa

```
import java.awt.Font;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

import edu.princeton.cs.algs4.StdDraw;
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class BondPercolation {

private static int size;
private int left = 0;
private int right;
private WeightedQuickUnionUF wquf; // ens serveix per pintar el cami de color
private WeightedQuickUnionUF wqufPerc; // aquesta ens servira per no confodre connectat amb ple
private boolean[] opened;
```

```
public BondPercolation(int n) {   // crea un reticle de nxn vrtexs , amb totes les 2(n^2-n) arestes tancades
if (n <= 0) throw new IllegalArgumentException("Given n <= 0");
size = n;
right = size*size + 1;
wquf = new WeightedQuickUnionUF(size*size + 2); // +2 pels dos vrtexs que afegim, dreta i esquerra
// es noms per la part grfica aquest
wqufPerc = new WeightedQuickUnionUF(size*size + 2);
opened = new boolean[2*(size*size - size)];

for (int i = 1; i <= size*size; i += size) {   // connectem tots els vrtexs de la primera columna amb el vrtex
      esquerra
wqufPerc.union(left, i);
wquf.union(left, i);
}

for (int i = size; i <= size*size; i+=size) { // connectem tots els vrtexs de l'ltima columna amb el vrtex dret
wqufPerc.union(right, i);
}
}


public void open(int x) {        // Obrim una aresta
opened[x-1] = true;
if (x <= size*size - size) {
int y = x + (int) ((x-1)/(size-1));
wqufPerc.union(y, y+1);
wquf.union(y, y+1);
}
else {
if (x % (size*size - size) == 0) x = size*size - size;
else x = x % (size*size - size);
wqufPerc.union(x, x+size);
wquf.union(x, x+size);
}

}


public boolean isOpen(int x) {
return opened[x-1];
}

public boolean isFullSite(int x) {
if (x > 0 && x <= size*size) {
return wquf.connected(left, x);
}
else {
throw new IndexOutOfBoundsException();
}
}


public boolean isFullBond(int x) {      //ens servir per saber si hem de pintar una aresta o no
if (x > 0 && x <= size*size - size) {
if (isFullSite(x + (int) ((x-1)/(size-1))) && isOpen(x)) return true;
else return false;
}
else if (x > size*size - size && x <= 2*(size*size - size)) {
int y = 0;
if (x % (size*size - size) == 0) y = size*size - size;
else y = x % (size*size - size);
if (isFullSite(y) && isOpen(x)) return true;
else return false;
}
else {
throw new IndexOutOfBoundsException();
}
}


public int numberOfOpenBonds() {
int comptador = 0;
for (int i = 1; i <= 2*(size*size - size); ++i) {
if (isOpen(i)) ++comptador;
}
return comptador;
}


public boolean percolates() {
return wqufPerc.connected(left, right);
}
```

```
//Visualitzador

private static final int DELAY = 1;
private static Scanner in;

public static void draw(BondPercolation perc, int n) {
StdDraw.clear();
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.setXscale(-0.05*n, 1.05*n);
StdDraw.setYscale(-0.05*n, 1.05*n); //deixem marge per escriure text
StdDraw.filledSquare(n/2.0, n/2.0, n/2.0);
double x = (double) 1/ (double) (5*n); // mida dels punts i arestes

// Dibuixem quadricula
for (int row = 1; row <= n; row++) {
for (int col = 1; col <= n; col++) {
StdDraw.setPenRadius(x);
StdDraw.setPenColor(StdDraw.WHITE);
StdDraw.point(col - 0.5, n - row + 0.5);
}
}

// Dibuixem arestes
int opened = 0;
StdDraw.setPenRadius(x/2);
StdDraw.setPenColor(StdDraw.WHITE);
for (int i = 1; i <= 2*(size*size - size); ++i) {
if (i <= size*size - size) {

if (perc.isFullBond(i)) {
int row = (int) (i-1) / (size - 1);
row ++;
int col = 0;
if (i % (size-1) == 0) col = size - 1;
else col = i % (size-1);
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
StdDraw.line(col - 0.5, n - row + 0.5, col - 0.5 + 1, n - row + 0.5);
opened ++;
}
else if (perc.isOpen(i)) {
int row = (int) (i-1) / (size - 1);
row ++;
int col = 0;
if (i % (size-1) == 0) col = size - 1;
else col = i % (size-1);
StdDraw.setPenColor(StdDraw.WHITE);
StdDraw.line(col - 0.5, n - row + 0.5, col - 0.5 + 1, n - row + 0.5);
opened ++;
}
}

if (i > size*size - size) {
if (perc.isFullBond(i)) {
int j = i - (size*size - size);
int row = (int) (j-1) / size;
row ++;
int col = 0;
if (j % size == 0) col = size;
else col = j % size;
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
StdDraw.line(col - 0.5, n - row + 0.5, col - 0.5, n - row + 0.5 - 1);
opened ++;
}
else if (perc.isOpen(i)) {
int j = i - (size*size - size);
int row = (int) (j-1) / size;
row ++;
int col = 0;
if (j % size == 0) col = size;
else col = j % size;
StdDraw.setPenColor(StdDraw.WHITE);
StdDraw.line(col - 0.5, n - row + 0.5, col - 0.5, n - row + 0.5 - 1);
opened ++;
}
}
}

// Escrivim el nombre d'arestes obertes i si percola o no a la part inferior del grfic
StdDraw.setFont(new Font("SansSerif", Font.PLAIN, 12));
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.text(0.25*n, -0.025*n, opened + " open bonds");
```

```
if (perc.percolates()) StdDraw.text(0.75*n, -0.025*n, "percolates");
else                   StdDraw.text(0.75*n, -0.025*n, "does not percolate");

}



public static void main(String[] args) throws FileNotFoundException {

// main consistent en obrir un fitxer
/*
BufferedReader fitxer = new BufferedReader(new FileReader
("C:\\Users\\PC\\workspace\\Percolation_and_visualizer\\Exemples\\prova.txt"));

in = new Scanner(fitxer);
int size = in.nextInt();
StdDraw.enableDoubleBuffering(); // engegem l'animaci
BondPercolation perc = new BondPercolation(size);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);

while (in.hasNextInt()) {
int x = in.nextInt();
perc.open(x);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);
}

if (perc.percolates()) StdOut.println("percola");
else StdOut.println("no percola");
*/


//main que s'obren arestes fins que percola

int size = StdIn.readInt();
StdDraw.enableDoubleBuffering();
BondPercolation perc = new BondPercolation(size);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);

while (!perc.percolates()) {
int x = StdRandom.uniform(2*(size*size - size))+1;
if (!perc.isOpen(x)) {
perc.open(x);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);
}
}

StdOut.println("probabilitat critica = " + (double) perc.numberOfOpenBonds() / (double) (2*(size*size - size)));

}

}
```

Segon programa:

```
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.StdStats;

public class MonteCarlo {
private int size;
private int trials;
private double[] criticprob;

public MonteCarlo(int n, int t) {    // executa t vegades el programa de percolaci en arestes per un reticle nxn
if (n <= 0 || t <= 0) {
throw new IllegalArgumentException("Given n <= 0 || t <= 0");
}
size = n;
trials = t;
criticprob = new double[trials];
for (int i = 0; i < trials; ++i) {
```

```
BondPercolation perc = new BondPercolation(size);
int comptador = 0;
while (!perc.percolates()) {
int x = StdRandom.uniform(2*(size*size - size))+1;
if (!perc.isOpen(x)) {
perc.open(x);
comptador++;
}
}
criticprob[i] = (double) comptador/(double) (2*(size*size-size));
StdOut.println("Fi de la vegada " + i);
}
}



public double mean() {
return StdStats.mean(criticprob);
}


public double stddev() {
return StdStats.stddev(criticprob);
}


public double confidenceLo() {
return mean()-(1.96*stddev())/Math.sqrt(trials);
}


public double confidenceHi() {
return mean()+(1.96*stddev())/Math.sqrt(trials);
}


public static void main(String[] args) {
long time_start, time_end;  // serveixen per calcular el temps que tarda el programa
int n = StdIn.readInt();
int t = StdIn.readInt();
time_start = System.currentTimeMillis();
MonteCarlo monte = new MonteCarlo(n, t);
time_end = System.currentTimeMillis();
StdOut.println("L'execucio del programa ha durat "+ ( time_end - time_start ) +" milisegons");
StdOut.println("Mitjana                 = " + monte.mean());
StdOut.println("Desviacio tipica        = " + monte.stddev());
StdOut.println("Interval confianca 95% = [" + monte.confidenceLo() + ", " + monte.confidenceHi() + "]");

}

}
```

5.3 Programes percolació en vèrtexs en el reticle triangular

Primer programa:

```
import java.awt.Font;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

import edu.princeton.cs.algs4.StdDraw;
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class SiteTriangularPercolation {

private int size;
private int left = 0;
private int right;
```

```
private WeightedQuickUnionUF wquf; // ens serveix per pintar el cami de color
private WeightedQuickUnionUF wqufPerc; // aquesta ens servira per no confodre connectat amb ple
private boolean [][] opened;


public SiteTriangularPercolation(int n) {    // creem el reticle nxn, amb tots els vrtexs tancats
if (n <= 0) throw new IllegalArgumentException("Given n <= 0");
size = n;
right = size*size + 1;
wquf = new WeightedQuickUnionUF(size*size + 2); // +2 per les dues arestes que afegim, dreta i esquerra
wqufPerc = new WeightedQuickUnionUF(size*size + 2);
opened = new boolean[size][size];
}


private int index(int row, int col) {  // serveix per passar de coordenades (row,col) a la posici del vector
return size*(row-1) + col;
}


public void open(int row, int col) {

opened[row-1][col-1] = true;
if (col == 1) {
wquf.union(index(row,col), left);
wqufPerc.union(index(row,col), left);
}
if (col == size) {
wqufPerc.union(index(row,col), right);
}
if (col > 1 && isOpen(row, col-1)) {    //mirem si podem connectar amb vrtex de l'esquerra
wquf.union(index(row, col-1), index(row, col));
wqufPerc.union(index(row, col-1), index(row, col));
}
if (col < size && isOpen(row, col+1)) {    //mirem si podem connectar amb vrtex de la dreta
wquf.union(index(row, col+1), index(row, col));
wqufPerc.union(index(row, col+1), index(row, col));
}
if (row > 1 && isOpen(row-1, col)) {    //mirem si podem connectar amb vrtex de "sobre" linies vermelles
wquf.union(index(row-1, col), index(row, col));
wqufPerc.union(index(row-1, col), index(row, col));
}
if (row < size && isOpen(row+1, col)) {    //mirem si podem connectar amb vrtex de "sota" linies vermelles
wquf.union(index(row+1, col), index(row, col));
wqufPerc.union(index(row+1, col), index(row, col));
}
if (row%2 == 0 && col > 1) {        //mirem les files parelles
if (isOpen(row-1,col-1)) {        //mirem diagonal cap a sobre l'esquerra
wquf.union(index(row-1, col-1), index(row,col));
wqufPerc.union(index(row-1, col-1), index(row,col));
}
if (row < size && isOpen(row+1,col-1)) {//mirem diagonal de sota l'esquerra
wquf.union(index(row+1, col-1), index(row,col));
wqufPerc.union(index(row+1, col-1), index(row,col));
}
}
else if (row%2 != 0 && col < size) {    //mirem les files senars
if (row > 1 && isOpen(row-1,col+1)) {
wquf.union(index(row-1,col+1), index(row,col));
wqufPerc.union(index(row-1,col+1), index(row,col));
}
if (row < size && isOpen(row+1,col+1)) {
wquf.union(index(row+1,col+1), index(row,col));
wqufPerc.union(index(row+1,col+1), index(row,col));
}
}
}


public boolean isOpen(int row, int col) {
return opened[row-1][col-1];
}


public boolean isFull(int row, int col) {  // serveix per la part grfica
if (row > 0 && row <= size && col > 0 && col <= size) {
return wquf.connected(left, index(row,col));
}
else {
throw new IndexOutOfBoundsException();
}
}
```

```
public int numberOfOpenSites() {
int comptador = 0;
for (int i = 1; i <= size; ++i) {
for (int j = 1; j <= size; ++j) {
if (isOpen(i,j)) ++comptador;
}
}
return comptador;
}


public boolean percolates() {
return wqufPerc.connected(left, right);
}


//Visualitzador
private static final int DELAY = 500;
private static Scanner in;

public static void draw(SiteTriangularPercolation perc, int n) {
++n;
StdDraw.clear();
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.setXscale(-0.05*n, 1.05*n);
StdDraw.setYscale(-0.05*n, 1.05*n); // deixem marge per escriure text
StdDraw.filledSquare(n/2.0, n/2.0, n/2.0);
--n;
double h = 0.5; //constant que sumare perqu la grfica quedi centrada de dalt a baix
double w = 0.25; //constant que sumare perqu la grfica quedi centrada d'esquerra a dreta

int opened = 0;
for (int row = 1; row <= n; row++) {
for (int col = 1; col <= n; col++) {
if (perc.isFull(row, col)) {
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
opened++;
}
else if (perc.isOpen(row, col)) {
StdDraw.setPenColor(StdDraw.WHITE);
opened++;
}
else StdDraw.setPenColor(StdDraw.BLACK);
if (row%2 == 0) {
double[] x = {col-0.5+w, col-0.05+w, col-0.05+w, col-0.5+w, col-1+0.05+w, col-1+0.05+w};
double[] y = {n-row+0.5+0.625-0.05+h, n-row+0.5+0.375-0.05+h, n-row+0.5-0.375+0.05+h,
      n-row+0.5-0.625+0.05+h, n-row+0.5-0.375+0.05+h, n-row+0.5+0.375-0.05+h};
StdDraw.filledPolygon(x, y);
}
else if (row%2 == 1) {
double[] x = {col+w, col+0.5-0.05+w, col+0.5-0.05+w, col+w, col-0.5+0.05+w, col-0.5+0.05+w};
double[] y = {n-row+0.5+0.625-0.05+h, n-row+0.5+0.375-0.05+h, n-row+0.5-0.375+0.05+h,
      n-row+0.5-0.625+0.05+h, n-row+0.5-0.375+0.05+h, n-row+0.5+0.375-0.05+h};
StdDraw.filledPolygon(x, y);
}
}
}

// Escrivim el nombre de vrtexs oberts i si percola o no a la part inferior del grfic
StdDraw.setFont(new Font("SansSerif", Font.PLAIN, 12));
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.text(0.25*n, -0.025*n, opened + " open sites");
if (perc.percolates()) StdDraw.text(0.75*n, -0.025*n, "percolates");
else                   StdDraw.text(0.75*n, -0.025*n, "does not percolate");

}



public static void main(String[] args) throws FileNotFoundException { // test client (optional)

/*
BufferedReader fitxer = new BufferedReader(new FileReader
("C:\\Users\\PC\\workspace\\Percolation_and_visualizer\\Exemples\\prova.txt"));

in = new Scanner(fitxer);
int size = in.nextInt();
StdDraw.enableDoubleBuffering(); //engegem l'animaci
SiteTriangularPercolation perc = new SiteTriangularPercolation(size);
draw(perc, size);
StdDraw.show();
```

```
StdDraw.pause(DELAY);

while (in.hasNextInt()) {
int row = in.nextInt();
int col = in.nextInt();
perc.open(row, col);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);

}

if (perc.percolates()) StdOut.println("percola");
else StdOut.println("no percola");
*/



int size = StdIn.readInt();
StdDraw.enableDoubleBuffering();
SiteTriangularPercolation perc = new SiteTriangularPercolation(size);
draw(perc, size);
StdDraw.show();
int comptador = 0;

while (!perc.percolates()) {
int row = StdRandom.uniform(size)+1;
int col = StdRandom.uniform(size)+1;
if (!perc.isOpen(row, col)) {
perc.open(row, col);
draw(perc, size);
StdDraw.show();
StdDraw.pause(DELAY);
comptador++;
}
}
StdOut.println("probabilitat critica= " + (double) comptador/(double) (size*size));

}

}
```

programa:

```
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.StdStats;

public class MonteCarlo {
private int size;
private int trials;
private double[] criticprob;



public MonteCarlo(int n, int t) {    // executa t vegades el programa de percolaci en vrtexs per un reticle
      triangular
if (n <= 0 || t <= 0) {
throw new IllegalArgumentException("Given n <= 0 || t <= 0");
}
size = n;
trials = t;
criticprob = new double[trials];
for (int i = 0; i < trials; ++i) {
SiteTriangularPercolation perc = new SiteTriangularPercolation(size);
int comptador = 0;
while (!perc.percolates()) {
int row = StdRandom.uniform(size)+1;
int col = StdRandom.uniform(size)+1;
if (!perc.isOpen(row, col)) {
perc.open(row, col);
comptador++;
}
}
criticprob[i] = (double) comptador/(double) (size*size);
StdOut.println("Fi de la vegada " + i + " \n");
}
}
```

```
public double mean() {
return StdStats.mean(criticprob);
}


public double stddev() {
return StdStats.stddev(criticprob);
}


public double confidenceLo() {
return mean()-(1.96*stddev())/Math.sqrt(trials);
}


public double confidenceHi() {
return mean()+(1.96*stddev())/Math.sqrt(trials);
}


public static void main(String[] args) {
long time_start, time_end;   // serveixen per calcular el temps que tarda el programa
int n = StdIn.readInt();
int t = StdIn.readInt();
time_start = System.currentTimeMillis();
MonteCarlo monte = new MonteCarlo(n, t);
time_end = System.currentTimeMillis();
StdOut.println("L'execucio del programa ha durat "+ ( time_end - time_start ) +" milisegons");
StdOut.println("Mitjana                = " + monte.mean());
StdOut.println("Desviacio tipica       = " + monte.stddev());
StdOut.println("Interval confianca 95% = [" + monte.confidenceLo() + ", " + monte.confidenceHi() + "]");

}

}
```

5.4 Programes percolació en arestes en el reticle triangular

Primer programa:

```
import java.awt.Font;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

import edu.princeton.cs.algs4.StdDraw;
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class BondTriangularPercolation {

private static int size;
private int left = 0;
private int right;
private WeightedQuickUnionUF wquf; // ens serveix per pintar el cami de color
private WeightedQuickUnionUF wqufPerc; // aquesta ens servira per no confodre connectat amb ple
private boolean[] opened;


public BondTriangularPercolation(int n) {// crea un reticle de nxn vrtexs , amb totes les 2(n^2-n)+(n-1)^2
      arestes tancades
if (n <= 0) throw new IllegalArgumentException("Given n <= 0");
size = n;
right = size*size + 1;
wquf = new WeightedQuickUnionUF(size*size + 2); // +2 pels dos vrtexs que afegim, dreta i esquerra
// es noms per la part grfica aquest
wqufPerc = new WeightedQuickUnionUF(size*size + 2);
opened = new boolean[2*(size*size - size) + (size-1)*(size-1)]; //vector per veure si les arestes estan obertes
```

```
for (int i = 1; i <= size*size; i += size) {  // connectem tots els  vrtexs  de la primera columna amb el vrtex
      esquerra
wqufPerc.union(left, i);
wquf.union(left, i);
}

for (int i = size; i <= size*size; i+=size) {  // connectem tots els  vrtexs  de l' ltima  columna amb el vrtex
      dret
wqufPerc.union(right, i);
}
}



public void open(int x) {        // Obrim una aresta
opened[x-1] = true;

if (x <= size*size - size) {
int y = x + (int) ((x-1)/(size-1));
//StdOut.print(x + " uneix " + y + " i " + (y+1) + "\n");
wqufPerc.union(y, y+1);
wquf.union(y, y+1);
}
else if (x > size*size - size && x <= 2*(size*size - size)) {
int y = x;
if (x % (size*size - size) == 0) x = size*size - size;   // aix  s  per l'aresta n^n - n, perqu  no sigui 0
else x = x % (size*size - size);
//StdOut.print(y + " uneix " + x + " i " + (x+size) + "\n");
wqufPerc.union(x, x+size);
wquf.union(x, x+size);
}
else {
int y = x;
x = x % (size*size - size);
int i = (int) (x-1)/(size-1);
if (i%2 == 0) {
int j = i;
wqufPerc.union(x+j, x+j+(size+1));
wquf.union(x+j, x+j+(size+1));
//StdOut.print(y + " uneix " + (x+j) + " i " + (x+j+(size+1)) + "\n");
}
else {
int j = i+1;
wqufPerc.union(x+j, x+j+(size-1));
wquf.union(x+j, x+j+(size-1));
//StdOut.print(y + " uneix " + (x+j) + " i " + (x+j+(size-1)) + "\n");
}

}


}


public boolean isOpen(int x) {   // l'aresta  est  oberta?
return opened[x-1];
}

public boolean isFullSite(int x) {   // is  site  (row, col)  full?
if (x > 0 && x <= size*size) {
return wquf.connected(left, x);
}
else {
throw new IndexOutOfBoundsException();
}
}


public boolean isFullBond(int x) {
if (x > 0 && x <= size*size - size) {
if (isFullSite(x + (int) ((x-1)/(size-1))) && isOpen(x)) return true; //mirem si el  vrtex  des d'on surt l'aresta
      est  ple
else return false;
}
else if (x > size*size - size && x <= 2*(size*size - size)) {
int y = 0;
if (x % (size*size - size) == 0) y = size*size - size;
else y = x % (size*size - size);
if (isFullSite(y) && isOpen(x)) return true;
else return false;
}
else if (x > 2*(size*size - size) && x <= 2*(size*size - size)+(size-1)*(size-1)) {
int y = x % (size*size - size);
```

```
int z = (int) (y-1)/(size-1);
y += z;
int j = x - 2*(size*size - size);                //Aquestes 4 linies segents sn per diferenciar
int row = (int) (j-1) / (size-1);                //els casos on l'aresta va d'esquerra a dreta baixant
row ++;                                        //o de dreta a esquerra baixant. Quan la fila s parella
if (row%2==0) ++y;                               //necessitem sumar 1 a l'index del vrtex d'on surt l'aresta
if ( isFullSite (y) && isOpen(x)) return true;
else return false;
}
else {
throw new IndexOutOfBoundsException();
}
}


public int numberOfOpenBonds() {
int comptador = 0;
for (int i = 1; i <= 2*(size*size - size) + (size-1)*(size-1); ++i) {
if (isOpen(i)) ++comptador;
}
return comptador;
}


public boolean percolates() {                // does the system percolate?
return wqufPerc.connected(left, right);
}




//Visualitzador
private static final int DELAY = 250;
private static Scanner in;

public static void draw(BondTriangularPercolation perc, int n) {
StdDraw.clear();
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.setXscale(-0.05*n, 1.05*n);
StdDraw.setYscale(-0.05*n, 1.05*n);  // leave a border to write text
StdDraw.filledSquare(n/2.0, n/2.0, n/2.0);
double x = (double) 1/ (double) (5*n);  // mida dels punts i arestes

// dibuixem quadricula
for (int row = 1; row <= n; row++) {
for (int col = 1; col <= n; col++) {
StdDraw.setPenRadius(x);
StdDraw.setPenColor(StdDraw.WHITE);
if (row%2==0) StdDraw.point(col - 0.75, n - row + 0.5);
else StdDraw.point(col - 0.25, n - row + 0.5);
}
}


// dibuixem arestes
int opened = 0;
StdDraw.setPenRadius(x/2);
StdDraw.setPenColor(StdDraw.WHITE);
for (int i = 1; i <= 2*(size*size - size) + (size-1)*(size-1); ++i) {
if (i <= size*size - size) {
if (perc.isFullBond(i)) {
int row = (int) (i-1) / (size - 1);
row ++;
int col = 0;
if (i % (size-1) == 0) col = size - 1;
else col = i % (size-1);
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
if (row%2 == 0) StdDraw.line(col - 0.75, n - row + 0.5, col - 0.75 + 1, n - row + 0.5);
else if (row%2 != 0) StdDraw.line(col - 0.25, n - row + 0.5, col - 0.25 + 1, n - row + 0.5);
opened ++;
}
else if (perc.isOpen(i)) {
int row = (int) (i-1) / (size - 1);
row ++;
int col = 0;
if (i % (size-1) == 0) col = size - 1;
else col = i % (size-1);
StdDraw.setPenColor(StdDraw.WHITE);
if (row%2 == 0) StdDraw.line(col - 0.75, n - row + 0.5, col - 0.75 + 1, n - row + 0.5);
else if (row%2 != 0) StdDraw.line(col - 0.25, n - row + 0.5, col - 0.25 + 1, n - row + 0.5);
opened ++;
```

```
}
}

else if (i > size*size - size && i <= 2*(size*size - size)) {
if (perc.isFullBond(i)) {
int j = i - (size*size - size);
int row = (int) (j-1) / size;
row++;
int col = 0;
if (j % size == 0) col = size;
else col = j % size;
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
if (row%2 == 0) StdDraw.line(col - 0.75, n - row + 0.5, col + 0.5 - 0.75, n - row + 0.5 - 1);
else StdDraw.line(col - 0.75, n - row - 0.5, col - 0.25, n - row + 0.5);
opened++;
}
else if (perc.isOpen(i)) {
int j = i - (size*size - size);
int row = (int) (j-1) / size;
row++;
int col = 0;
if (j % size == 0) col = size;
else col = j % size;
StdDraw.setPenColor(StdDraw.WHITE);
if (row%2 == 0) StdDraw.line(col - 0.75, n - row + 0.5, col + 0.5 - 0.75, n - row + 0.5 - 1);
else StdDraw.line(col - 0.75, n - row - 0.5, col - 0.25, n - row + 0.5);
opened++;
}
}

else if (i > 2*(size*size - size) && i <= 2*(size*size - size)+(size-1)*(size-1)) {
if (perc.isFullBond(i)) {
int j = i - 2*(size*size - size);
int row = (int) (j-1) / (size-1);
row++;
int col = 0;
if (j % (size-1) == 0) col = size-1;
else col = j % (size-1);
if (row%2 == 0) ++col;
StdDraw.setPenColor(StdDraw.BOOK_LIGHT_BLUE);
if (row%2 == 0) StdDraw.line(col - 1.25, n - row - 0.5, col - 0.75, n - row + 0.5);
else StdDraw.line(col - 0.25, n - row + 0.5, col + 0.5 - 0.25, n - row + 0.5 - 1);
opened++;
}
else if (perc.isOpen(i)) {
int j = i - 2*(size*size - size);
int row = (int) (j-1) / (size-1);
row++;
int col = 0;
if (j % (size-1) == 0) col = size-1;
else col = j % (size-1);
if (row%2 == 0) ++col;
StdDraw.setPenColor(StdDraw.WHITE);
if (row%2 == 0) StdDraw.line(col - 1.25, n - row - 0.5, col - 0.75, n - row + 0.5);
else StdDraw.line(col - 0.25, n - row + 0.5, col + 0.5 - 0.25, n - row + 0.5 - 1);
opened++;
}
}

}

// Escrivim el nombre d'arestes obertes i si percola o no a la part inferior del grfic
StdDraw.setFont(new Font("SansSerif", Font.PLAIN, 12));
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.text(0.25*n, -0.025*n, opened + " open bonds");
if (perc.percolates()) StdDraw.text(0.75*n, -0.025*n, "percolates");
else                   StdDraw.text(0.75*n, -0.025*n, "does not percolate");

}



public static void main(String[] args) throws FileNotFoundException {

// main consistent en obrir un fitxer
/*
BufferedReader fitxer = new BufferedReader(new FileReader
("C:\\Users\\PC\\workspace\\Percolation_and_visualizer\\Exemples\\prova.txt"));

in = new Scanner(fitxer);
int size = in.nextInt();
StdDraw.enableDoubleBuffering();
```

```
BondTriangularPercolation tperc = new BondTriangularPercolation(size);
draw(tperc, size);
StdDraw.show();
StdDraw.pause(DELAY);

while (in.hasNextInt()) {
int x = in.nextInt();
tperc.open(x);
draw(tperc, size);
StdDraw.show();
StdDraw.pause(DELAY);
}

if (tperc.percolates()) StdOut.println("percola");
else StdOut.println("no percola");
*/


//main que s'obren arestes fins que percola

int size = StdIn.readInt();
StdDraw.enableDoubleBuffering();
BondTriangularPercolation tperc = new BondTriangularPercolation(size);
draw(tperc, size);
StdDraw.show();
StdDraw.pause(DELAY);

while (!tperc.percolates()) {
int x = StdRandom.uniform(2*(size*size - size) + (size-1)*(size-1))+1;
if (!tperc.isOpen(x)) {
tperc.open(x);
draw(tperc, size);
StdDraw.show();
StdDraw.pause(DELAY);
}
}

StdOut.println("probabilitat critica = " + (double) tperc.numberOfOpenBonds() / (double) (2*(size*size -
      size)+(size-1)*(size-1)));


}

}
```

Segon programa:

```
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.StdStats;

public class TriangularMonteCarlo {
private int size;
private int trials;
private double[] criticprob;



public TriangularMonteCarlo(int n, int t) {    // executa t vegades el programa de percolaci en arestes per un
      reticle triangular
if (n <= 0 || t <= 0) {
throw new IllegalArgumentException("Given n <= 0 || t <= 0");
}
size = n;
trials = t;
criticprob = new double[trials];
for (int i = 0; i < trials; ++i) {
BondTriangularPercolation perc = new BondTriangularPercolation(size);
int comptador = 0;
while (!perc.percolates()) {
int x = StdRandom.uniform(2*(size*size - size)+(size-1)*(size-1))+1;
if (!perc.isOpen(x)) {
perc.open(x);
comptador++;
}
}
criticprob[i] = (double) comptador/(double) (2*(size*size-size)+(size-1)*(size-1));
StdOut.println("Fi de la vegada " + i + "\n");
}
}
```

```
public double mean() {
return StdStats.mean(criticprob);
}


public double stddev() {
return StdStats.stddev(criticprob);
}


public double confidenceLo() {
return mean()-(1.96*stddev())/Math.sqrt(trials);
}


public double confidenceHi() {
return mean()+(1.96*stddev())/Math.sqrt(trials);
}


public static void main(String[] args) {
long time_start, time_end; // serveixen per calcular el temps que tarda el programa
int n = StdIn.readInt();
int t = StdIn.readInt();
time_start = System.currentTimeMillis();
TriangularMonteCarlo monte = new TriangularMonteCarlo(n, t);
time_end = System.currentTimeMillis();
StdOut.println("L'execucio del programa ha durat "+ ( time_end - time_start ) +" milisegons");
StdOut.println("Mitjana                  = " + monte.mean());
StdOut.println("Desviacio tipica         = " + monte.stddev());
StdOut.println("Interval confianca 95% = [" + monte.confidenceLo() + ", " + monte.confidenceHi() + "]");
}

}
```

5.5 Programes percolació en vèrtexs en $mathbbZ^3$

Primer programa:

```
import java.awt.Font;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

import edu.princeton.cs.algs4.StdDraw;
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class Z3SitePercolation {

private int size;
private int left = 0;
private int right;
private WeightedQuickUnionUF wquf; // ens serveix per pintar el cami de color
private WeightedQuickUnionUF wqufPerc; // aquesta ens servira per no confodre connectat amb ple
private boolean [][][] opened;
private static Scanner in;


public Z3SitePercolation(int n) {          // creem el reticle nxnxn, amb tots els vrtexs tancats
if (n <= 0) throw new IllegalArgumentException("Given n <= 0");
size = n;
right = size*size*size + 1;
wqufPerc = new WeightedQuickUnionUF(size*size*size + 2);
opened = new boolean[size][size][size];
}


private int index(int row, int col, int prof) { //serveix per passar de coordenades (row,col,prof) a la posici del
      vector
for (int i = 1; i <= size; ++i) {
```

```
if (prof == i) return size*(row-1) + col + (prof-1)*size*size;
}
return 0;
}


public void open(int row, int col, int prof) {

opened[row-1][col-1][prof-1] = true;
if (col == 1) {
wqufPerc.union(index(row,col,prof), left);
}
if (col == size) {
wqufPerc.union(index(row,col,prof), right);
}


if (col > 1 && isOpen(row, col-1, prof)) {        //mirem si podem connectar amb vrtex de l'esquerra de la
    mateixa cara
wqufPerc.union(index(row, col-1, prof), index(row, col, prof));
}
if (col < size && isOpen(row, col+1, prof)) {     //mirem si podem connectar amb vrtex de la dreta de la mateixa
    cara
wqufPerc.union(index(row, col+1, prof), index(row, col, prof));
}
if (row > 1 && isOpen(row-1, col, prof)) {    //mirem si podem connectar amb vrtex de sobre de la mateixa cara
wqufPerc.union(index(row-1, col, prof), index(row, col, prof));
}
if (row < size && isOpen(row+1, col, prof)) {     //mirem si podem connectar amb vrtex de sota de la mateixa cara
wqufPerc.union(index(row+1, col, prof), index(row, col, prof));
}


if (prof > 1 && isOpen(row, col, prof-1)) {          //mirem si podem connectar amb vrtex de la cara del davant
wqufPerc.union(index(row, col, prof-1), index(row, col, prof));
}
if (prof < size && isOpen(row, col, prof+1)) {          //mirem si podem connectar amb vrtex de la cara del darrera
wqufPerc.union(index(row, col, prof+1), index(row, col, prof));
}



}


public boolean isOpen(int row, int col, int prof) {
return opened[row-1][col-1][prof-1];
}


public boolean isFull(int row, int col, int prof) {
if (row > 0 && row <= size && col > 0 && col <= size) {
return wquf.connected(left, index(row,col, prof));
}
else {
throw new IndexOutOfBoundsException();
}
}


public int numberOfOpenSites() {
int comptador = 0;
for (int i = 1; i <= size; ++i) {
for (int j = 1; j <= size; ++j) {
for (int k = 0; k <= size; ++k) {
if (isOpen(i,j,k)) ++comptador;
}
}
}
return comptador;
}


public boolean percolates() {               // does the system percolate?
return wqufPerc.connected(left, right);
}



public static void main(String[] args) throws FileNotFoundException {  // test client (optional)
BufferedReader fitxer = new BufferedReader(new FileReader
("C:\\Users\\PC\\workspace\\Percolation_and_visualizer\\Exemples\\prova.txt"));
```

```
in = new Scanner(fitxer);
int size = in.nextInt();
Z3SitePercolation perc = new Z3SitePercolation(size);

while (in.hasNextInt()) {
int row = in.nextInt();
int col = in.nextInt();
int prof = in.nextInt();
perc.open(row, col, prof);
}

if (perc.percolates()) StdOut.println("percola");
else StdOut.println("no percola");


}

}
```

Segon programa:

```
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.StdStats;

public class MonteCarlo {

private int size;
private int trials;
private double[] criticprob;


public MonteCarlo(int n, int t) {    // executa t vegades el programa de percolaci en vrtexs per un reticle nxnxn
if (n <= 0 || t <= 0) {
throw new IllegalArgumentException("Given n <= 0 || t <= 0");
}
size = n;
trials = t;
criticprob = new double[trials];
for (int i = 0; i < trials; ++i) {
Z3SitePercolation perc = new Z3SitePercolation(size);
int comptador = 0;
while (!perc.percolates()) {
int row = StdRandom.uniform(size)+1;
int col = StdRandom.uniform(size)+1;
int prof = StdRandom.uniform(size)+1;
if (!perc.isOpen(row, col, prof)) {
perc.open(row, col, prof);
comptador++;
}
}
criticprob[i] = (double) comptador/(double) (size*size*size);
StdOut.println("fi de la vegada " + i);
}
}


public double mean() {
return StdStats.mean(criticprob);
}


public double stddev() {
return StdStats.stddev(criticprob);
}


public double confidenceLo() {
return mean()-(1.96*stddev())/Math.sqrt(trials);
}


public double confidenceHi() {
return mean()+(1.96*stddev())/Math.sqrt(trials);
}



public static void main(String[] args) {
```

```
long time_start, time_end; // serveixen per calcular el temps que tarda el programa
int n = StdIn.readInt();
int T = StdIn.readInt();
time_start = System.currentTimeMillis();
MonteCarlo monte = new MonteCarlo(n, T);
time_end = System.currentTimeMillis();
StdOut.println("L'execucio del programa ha durat "+ ( time_end - time_start ) +" milisegons");
StdOut.println("Mitjana                  = " + monte.mean());
StdOut.println("Desviacio tipica         = " + monte.stddev());
StdOut.println("Interval confianca 95% = [" + monte.confidenceLo() + ", " + monte.confidenceHi() + "]");
}

}
```

5.6 Programes percolació en arestes en $\mathbb{Z}^3$

Primer programa:

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class Z3BondPercolation {

private static int size;
private int left = 0;
private int right;
private WeightedQuickUnionUF wqufPerc; // aquesta ens servira per no confodre connectat amb ple
private boolean[] opened;


public Z3BondPercolation(int n) {   // crea un reticle de nxnxn vrtexs, amb totes les arestes tancades
if (n <= 0) throw new IllegalArgumentException("Given n <= 0");
size = n;
right = size*size*size + 1;
wqufPerc = new WeightedQuickUnionUF(size*size*size + 2);
opened = new boolean[3*(size*size*size - size*size)];


for (int i = 1; i <= size*size*size; i += size) { //connectem vrtexs del costat esquerra amb el vrtex imaginari
    esquerra
wqufPerc.union(left, i);
}

for (int i = size; i <= size*size*size; i += size) { //connectem vrtexs del costat dret amb el vrtex imaginari
    dret
wqufPerc.union(right, i);
}

}


public void open(int x) {       // obrim una aresta
opened[x-1] = true;

if (x > 0 && x <= 2*(size*size*size - size*size)) {
int j = 0;
for (int i = 1; i < 2*size; i += 2) {
if (x > (i-1)*(size*size - size) && x <= i*(size*size-size)) {
int z = x%(2*(size*size-size));
int y = z + (int) ((z-1)/(size-1));
//StdOut.print(x + " uneix " + (y+j*size*size) + " i " + (y+j*size*size+1) + "\n");
wqufPerc.union(y+j*size*size, y+j*size*size+1);
return;
}
else if (x > i*(size*size - size) && x <= (i+1)*(size*size - size)) {
int z = x%(2*(size*size-size));
int y = x;
if (z % (size*size - size) == 0) z = size*size - size;
else z = z % (size*size - size);
```

```
//StdOut.print(y + " uneix " + (z+j*size*size) + " i " + (z+j*size*size+size) + "\n");
wqufPerc.union(z+j*size*size, z+j*size*size+size);
return;
}
else ++j;
}
}
if (x > 2*(size*size*size - size*size)) {
int z = x % (2*(size*size*size - size*size));
//StdOut.print(x + " uneix " + z + " i " + (z+size*size) + "\n");
wqufPerc.union(z,z+size*size);
return;
}
}


public boolean isOpen(int x) {   // is site (row, col) open?
return opened[x-1];
}


public int numberOfOpenBonds() {
int comptador = 0;
for (int i = 1; i <= 3*(size*size*size - size*size); ++i) {
if (isOpen(i)) ++comptador;
}
return comptador;
}


public boolean percolates() {               // does the system percolate?
return wqufPerc.connected(left, right);
}



public static void main(String[] args) throws FileNotFoundException {  // test client (optional)

// main consistent en obrir un fitxer
/*
BufferedReader fitxer = new BufferedReader(new FileReader
("C:\\Users\\PC\\workspace\\Percolation_and_visualizer\\Exemples\\prova.txt"));

Scanner in = new Scanner(fitxer);
int size = in.nextInt();
Z3BondPercolation perc = new Z3BondPercolation(size);

while (in.hasNextInt()) {
int x = in.nextInt();
perc.open(x);
}

if (perc.percolates()) StdOut.println("percola");
else StdOut.println("no percola");
*/


//main que s'obren arestes fins que percola

int size = StdIn.readInt();
Z3BondPercolation perc = new Z3BondPercolation(size);
while (!perc.percolates()) {
int x = StdRandom.uniform(3*(size*size*size - size*size))+1;
if (!perc.isOpen(x)) {
perc.open(x);
}
}

StdOut.println("probabilitat critica = " + (double) perc.numberOfOpenBonds() / (double) (3*(size*size*size -
      size*size)));

}

}
```

Segon programa:

```
import edu.princeton.cs.algs4.StdIn;
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.StdStats;
```

```
public class MonteCarlo {

private int size;
private int trials;
private double[] criticprob;


public MonteCarlo(int n, int t) {    // executa t vegades el programa de percolaci en arestes per un reticle nxnxn
if (n <= 0 || t <= 0) {
throw new IllegalArgumentException("Given n <= 0 || t <= 0");
}
size = n;
trials = t;
criticprob = new double[trials];
for (int i = 0; i < trials; ++i) {
Z3BondPercolation perc = new Z3BondPercolation(size);
int comptador = 0;
while (!perc.percolates()) {
int x = StdRandom.uniform(3*(size*size*size-size*size))+1;
if (!perc.isOpen(x)) {
perc.open(x);
comptador++;
}
}
criticprob[i] = (double) comptador/(double) (3*(size*size*size-size*size));
StdOut.println("fi de la vegada " + i);
}
}


public double mean() {
return StdStats.mean(criticprob);
}


public double stddev() {
return StdStats.stddev(criticprob);
}


public double confidenceLo() {
return mean()-(1.96*stddev())/Math.sqrt(trials);
}


public double confidenceHi() {
return mean()+(1.96*stddev())/Math.sqrt(trials);
}



public static void main(String[] args) {
long time_start, time_end; // serveixen per calcular el temps que tarda el programa
int n = StdIn.readInt();
int T = StdIn.readInt();
time_start = System.currentTimeMillis();
MonteCarlo monte = new MonteCarlo(n, T);
time_end = System.currentTimeMillis();
StdOut.println("L'execucio del programa ha durat "+ ( time_end - time_start ) +" milisegons");
StdOut.println("Mitjana                = " + monte.mean());
StdOut.println("Desviacio tipica       = " + monte.stddev());
StdOut.println("Interval confianca 95% = [" + monte.confidenceLo() + ", " + monte.confidenceHi() + "]");
}

}
```

Bibliografia

[1] Broadbent, S. R.; Hammersley, J. M.Percolation processes. I. Crystals and mazes. Proc. Cambridge Philos. Soc. 53 (1957), 629–641.

[2] Kesten, Harry The critical probability of bond percolation on the square lattice equals 12. Comm. Math. Phys. 74 (1980), no. 1, 41–59.

[3] Harris, T. E. A lower bound for the critical probability in a certain percolation process. Proc. Cambridge Philos. Soc. 56 1960 13–20.

[4] Russo, Lucio A note on percolation. Z. Wahrscheinlichkeitstheorie und Verw. Gebiete 43 (1978), no. 1, 39–48.

[5] Seymour, P. D.; Welsh, D. J. A. Percolation probabilities on the square lattice. Advances in graph theory (Cambridge Combinatorial Conf., Trinity College, Cambridge, 1977). Ann. Discrete Math. 3 (1978), 227–245.

[6] Bollobás, Béla; Riordan, Oliver Percolation. Cambridge University Press, New York, 2006.

[7] https://www.coursera.org/learn/algorithms-part1

[8] Margulis, G. A. Probabilistic characteristics of graphs with large connectivity. (Russian) Problemy Peredači Informacii 10 (1974), no. 2, 101–108.

[9] Friedgut, Ehud(IL-HEBR-IM); Kalai, Gil(IL-HEBR-IM). Every monotone graph property has a sharp threshold. (English summary) Proc. Amer. Math. Soc. 124 (1996), no. 10, 2993–3002.

[10] Bollobás, Béla; Riordan, Oliver A short proof of the Harris-Kesten theorem. Bull. London Math. Soc. 38 (2006), no. 3, 470–484.

[11] Menshikov, M. V.(2-MOSC) Coincidence of critical points in percolation problems. Dokl. Akad. Nauk SSSR 288 (1986), no. 6, 1308–1311.

[12] Russo, Lucio On the critical percolation probabilities. Z. Wahrsch. Verw. Gebiete 56 (1981), no. 2, 229–237.

[13] Sykes, M. F.; Essam, J. W. Exact critical percolation probabilities for site and bond problems in two dimensions. J. Mathematical Phys. 5 1964 1117–1127.

[14] Aizenman, M.; Kesten, H.; Newman, C. M. Uniqueness of the infinite cluster and continuity of connectivity functions for short and long range percolation. Comm. Math. Phys. 111 (1987), no. 4, 505–531.

[15] Burton, R. M.; Keane, M. Density and uniqueness in percolation. Comm. Math. Phys. 121 (1989), no. 3, 501–505.

[16] Kesten, Harry. Percolation theory for mathematicians. Progress in Probability and Statistics, 2. Birkhäuser, Boston, Mass., 1982.

[17] http://coursera.cs.princeton.edu/algs4/assignments/percolation.html

[18] Jacobsen, J. L. (2015). Çritical points of Potts and O(N) models from eigenvalue identities in periodic Temperley-Lieb algebras". Preprint. 48: 454003.

[19] Xu, Xiao; Junfeng Wang, Jian-Ping Lv, Youjin Deng (2014). "Simultaneous analysis of three-dimensional percolation models". Frontiers of Physics. 9 (1): 113–119.

[20] Wang, J; Z. Zhou; W. Zhang; T. Garoni; Y. Deng (2013). "Bond and site percolation in three dimensions". Physical Review E. 87.